T.C.
MİLLÎ EĞİTİM BAKANLIĞI

YENİLENEBİLİR ENERJİ TEKNOLOJİLERİ

MİKRODENETLEYİCİLERLE PROGRAMLAMA VE PROJE UYGULAMALARI

Ankara, 2015

- Bu modül, mesleki ve teknik eğitim okul/kurumlarında uygulanan Çerçeve Öğretim Programlarında yer alan yeterlikleri kazandırmaya yönelik olarak öğrencilere rehberlik etmek amacıyla hazırlanmış bireysel öğrenme materyalidir.
- Millî Eğitim Bakanlığınca ücretsiz olarak verilmiştir.
- PARA İLE SATILMAZ.

İÇİNDEKİLER

AÇIKLAMALAR iii
GİRİŞ 1
ÖĞRENME FAALİYETİ–1 3
1. C DERLEYiCiSi 3
1.1. C Derleyicisini Tanımak 3
1.1.1. C Programlama Dillerinin Özellikleri 3
1.1.2. C Programlama Dili İle Kullanılan Programın Özellikleri ve Tanıtımı 4
1.2. DERLEYİCİ C Derleyicisi Temel Kavramları 13
1.2.1. DERLEYİCİ C Dilinde Kullanılan Operatörler ve Örnek Programlar 13
1.3. DERLEYİCİ C ile Program Derlemek ve Denemek 20
1.3.1. Diziler 20
1.3.2. İşaretçiler (Pointer) 21
1.3.3. Yapılar ve Fonksiyonlarla Program Yazma 22
1.3.4. 16F ve 18F Serisi İşlemcilerle Program Yazılımı 23
1.3.5. Derleyici Kullanımı 23
UYGULAMA FAALİYETİ 29
ÖLÇME VE DEĞERLENDİRME 34
ÖĞRENME FAALİYETİ–2 35
2. DERLEYİCİ C İLE UYGULAMALARI 35
2.1. Port Giriş/Çıkış İşlemleri ve Örnek Program Yazılımları 35
2.1.1. SET_TRIS_X( ) Komutu 35
2.1.2. GET_TRIS_X( ) Komutu 36
2.1.3. OUTPUT_X( ) Komutu 36
2.1.4. OUTPUT_HIGH( ) Komutu 36
2.1.5. OUTPUT_LOW( ) Komutu 37
2.1.6. OUTPUT_TOGGLE( ) Komutu 37
2.1.7. OUTPUT_BIT( ) Komutu 37
2.1.8. INPUT_X( ) Komutu 37
2.1.9. INPUT( ) Komutu 37
2.1.10. INPUT_SATATE( ) Komutu 38
2.1.11. Örnek Programlar 38
2.2. Karar Verme Döngüleri ve Kullanım Örnekleri 41
2.2.1. IF Komutu 41
2.2.2. IF-ELSE Komutu 41
2.2.3. SWITCH – CASE Komutu 43
2.2.4. WHILE Komutu 45
2.2.5. DO-WHILE Komutu 46
2.2.6. FOR Komutu 48
2.3. Gecikme Fonksiyonları ve Örnek Program Yazılımları 48
2.3.1. DELAY_US( ) Komutu 48
2.3.2. DELAY_MS( ) Komutu 49
2.3.3. DELAY_CYCLES( ) Komutu 49
2.4. Port Çoğullama ve Matris Display Uygulamaları 49
2.4.1. 74138 Entegresi ile Port Çoğullama 49
2.4.2. Matris Display Uygulamaları 53

2.5. Kesme (interrupt) İşlemleri 55
2.6. Zamanlayıcı ve Sayıcı İşlemleri 58
2.6.1. TIMER0 Birimi 58
2.6.2. TIMER1 Birimi 59
2.6.3. TIMER2 Birimi 60
2.7. Kesme ve Timer ile İlgili Örnek Yazılım Geliştirme 60
2.8. Capture/Compare/Pwm Biriminin Çalıştırılması 63
2.8.1. Capture (Yakalama) Modu 63
2.8.2. Compare (Karşılaştırma) Modu 64
2.8.3. PWM Modu 64
2.8.4. SETUP_CCPx( ) Komutu 64
2.9. PWM Uygulamaları 67
2.9.1. PWM Sinyalin Görev Çevrim Süresini Değiştirme 67
2.9.2. PWM Sinyalin Frekansını Değiştirme 69
UYGULAMA FAALİYETİ 71
ÖLÇME VE DEĞERLENDİRME 73
MODÜL DEĞERLENDİRME 74
CEVAP ANAHTARLARI 75
KAYNAKÇA 77

AÇIKLAMALAR

ALAN	**Yenilenebilir Enerji Teknolojileri**
DAL/MESLEK	**Rüzgâr Enerjisi Sistemleri/Güneş Enerji Sistemleri**
MODÜLÜN ADI	**Mikrodenetleyicilerle Programlama ve Proje Uygulamaları**
MODÜLÜN TANIMI	Mikrodenetleyicilerle program geliştirebilmek
SÜRE	40/32
ÖN KOŞUL	Bu modülün ön koşulu yoktur.
YETERLİK	Mikrodenetleyici programlama ve proje uygulamaları yapmak
MODÜLÜN AMACI	**Genel Amaç** Güneş/Rüzgâr santrali otomasyonu atölyesi ortamı sağlandığında, mikrodenetleyici programlama ve proje uygulamaları yapılabileceksiniz. **Amaçlar** **1.** CCS derleyicisini tanıyabilecek ve programı derleyebileceksiniz. **2.** CCS C ile basit uygulamalar yapabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	Bilgisayar, mikrodenetleyici programlama yazılımı
ÖLÇME VE DEĞERLENDİRME	Modül içinde yer alan her öğrenme faaliyetinden sonra verilen ölçme araçları ile kendinizi değerlendireceksiniz. Öğretmen, modül sonunda ölçme aracı (çoktan seçmeli test, doğru-yanlış testi, boşluk doldurma, eşleştirme vb.) kullanarak modül uygulamaları ile kazandığınız bilgi ve becerileri ölçerek sizi değerlendirecektir.

GİRİŞ

Sevgili Öğrenci,

Yüksek bütçeli yatırımların yapıldığı enerji sistemlerinin maksimum performansta çalışması ve herhangi bir sorun oluştuğunda kaynağının hızlı bir şekilde tespit edilmesi için sensörler aracılığıyla birçok değişkenin kontrol edilmesi ve sistemin ilgili birimleri arasında hızlı bir veri iletişiminin sağlanması gerekmektedir.

Günümüz teknolojisinde bu gibi işlemler mikroişlemcili ya da mikrodenetleyicili sistemler aracılığıyla yapılmaktadır.

Mikrodenetleyicilerin elde edilmesi, programlanması ve devre uygulamaları daha basit olduğundan, bu modül size C dilini kullanarak mikrodenetleyiciler için program yazma ve uygulama geliştirme yeteneği kazandıracaktır.

ÖĞRENME FAALİYETİ–1

AMAÇ

DERLEYİCİ C derleyicisini tanıyacak ve programı derleyebileceksiniz.

ARAŞTIRMA

- Piyasada rahat bulunabilen denetleyici türlerini araştırınız.
- Mikrodenetleyici programlamada hangi dillerin ve derleyicilerin kullanıldığını araştırınız.
- Mikrodenetleyici programlamak için neler gerektiğini araştırınız.

Bu araştırmaları yapmak için çevrenizde ya da internet üzerinde elektronik malzeme satışı yapan ya da mikro denetleyici programcılığıyla uğraşan kişilerden ve mikrodenetleyiciler üzerine yazılan kitaplardan faydalanabilirsiniz.

1. C DERLEYİCİSİ

1.1. C Derleyicisini Tanımak

1.1.1. C Programlama Dillerinin Özellikleri

1970'li yılların başında Dennis Ritchie tarafından geliştirilen C programlama dili başlangıçta UNIX sistemler için geliştirilmiş olmasına rağmen günümüzde en fazla tercih edilen programlama dilleri arasına girmiş ve 1983 yılında ANSI (Amerikan Ulusal Standart Enstitüsü) tarafından standardize edildikten sonra yüksek oranda taşınabilir bir sistem programlama dili olmuştur.

Assembly kadar alt düzeye (makine diline) inebilen C programlama dili ifadeleri ve yazılış biçimleri açısından günlük İngilizce diline de çok benzemektedir. Bu özelliği ile de kolay anlaşılıp öğrenilebilen C programlama dili yüksek seviye dillerine de benzemektedir. Bu benzerlikler nedeniyle C programlama dili orta düzey programlama dili olarak isimlendirilmektedir.

C derleyicisi ANSI C'nin temel tanımlamalarının büyük çoğunluğunu kabul eder ve diğer C temelli mikrodenetleyici derleyicilerine göre C dili ile daha fazla uyum içindedir.

1.1.2. C Programlama Dili İle Kullanılan Programın Özellikleri ve Tanıtımı

1.1.2.1. DERLEYİCİ C Programının Özellikleri

DERLEYİCİ C program için DERLEYİCİ firması tarafından sunulan PIC C program kullanılacaktır.

DERLEYİCİ C derleyicisi farklı birimlerden oluşur ve kurulum yaparken satın alınan birimlerin hepsi ya da istenenleri kuruluma dâhil edilebilir. Bu birimler;

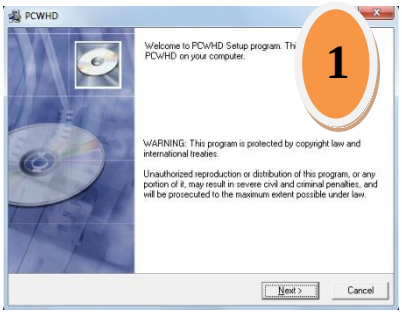
- PCB Compiler : 12 bit komut boyuna sahip mikrodenetleyiciler için,
- PCM Compiler : 14 bit komut boyuna sahip mikrodenetleyiciler için,
- PCH Compiler : 16 bit komut boyuna sahip mikrodenetleyiciler için,
- PCD Compiler : 24 bit komut boyuna sahip mikrodenetleyiciler için,

  kullanılır. Ancak bunların birkaçını barındıran derleyici tanımlarıda mevcuttur. Bunlar;
- PCW : PCB ve PCM derleyicilerini,
- PCWH : PCB, PCM ve PCH derleyicilerini,
- PCWHD : PCB, PCM, PCH ve PCD derleyicilerini barındırır.

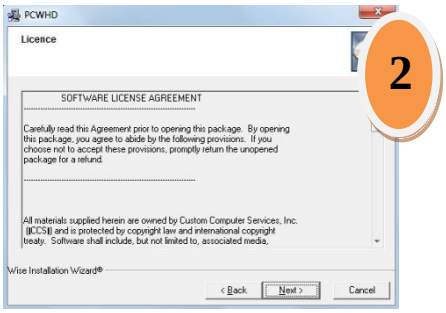
1.1.2.2. DERLEYİCİ C Programının Kurulumu

Demo sürüm için doldurulması gereken bir form sayfası ile karşılaşacaksınız. Sayfayı doldurun ve altta bulunan "Submit" butonuna tıkladığınızda isminizin ve @-posta adresinizin yazılacağı ekranla karşılaşacaksınız. Bu bölüm doldurulup "Download Demo" butonuna tıklandığında programın en son sürümü demo olarak bilgisayarınıza indirilir.

Kurulum dosyası üzerine fare ile çift tıklandığında kurulum açılış ekranı karşımıza gelir ve sürekli olarak "NEXT" butonuna tıklanır. Son olarak "FINISH" butonunun görüldüğü kurulum bitiş ekranı karşımıza gelir. "FINISH" butonuna tıklanarak kurulum sonlandırılır. Aşağıda kurulum aşamaları sırasıyla gösterilmiştir.

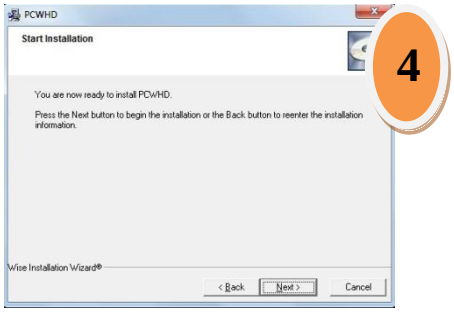


Resim 1.1: DERLEYİCİ C Kurulum açılış ekranı



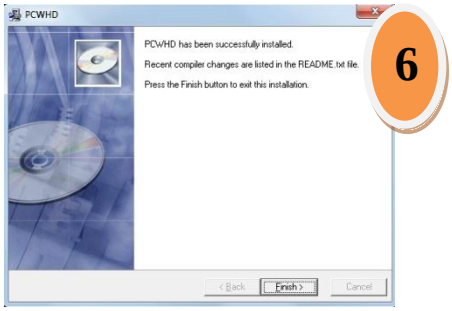
Resim 1.2: DERLEYİCİ C Lisans sözleşme ekranı

Resim 1.3: Dizin seçme ekranı



Resim 1.4: Kuruluma başlama ekranı

Resim 1.5: Dosyalar bilgisayara kurulumu

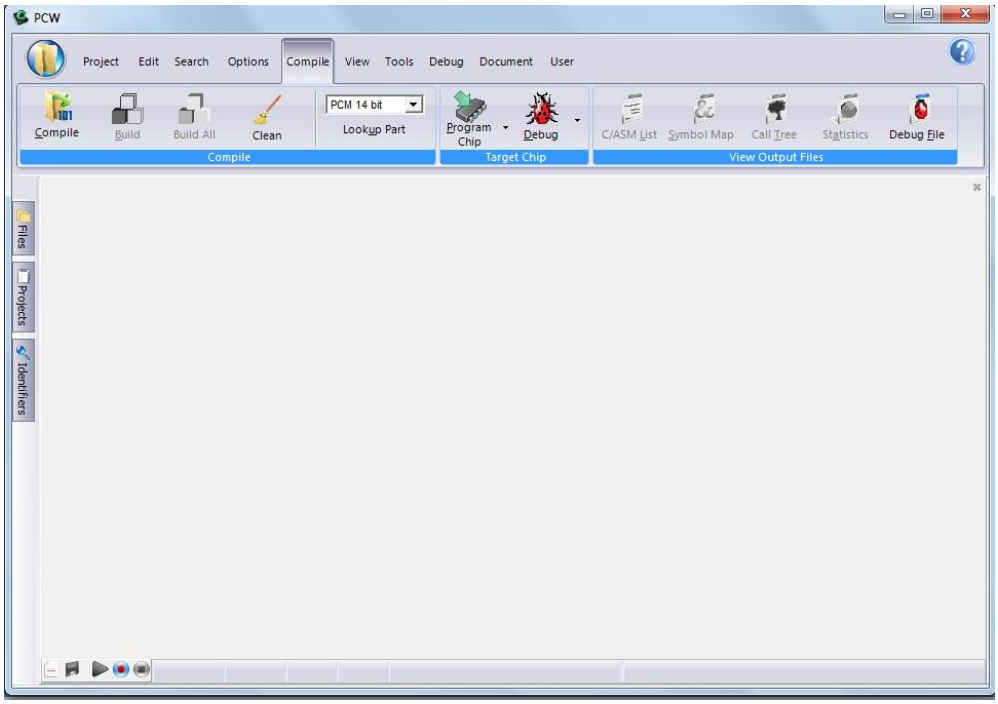


Resim 1.6: Kurulum bitiş ekranı

1.1.2.3. DERLEYİCİ C Derleyici Arayüzü ve Menüleri

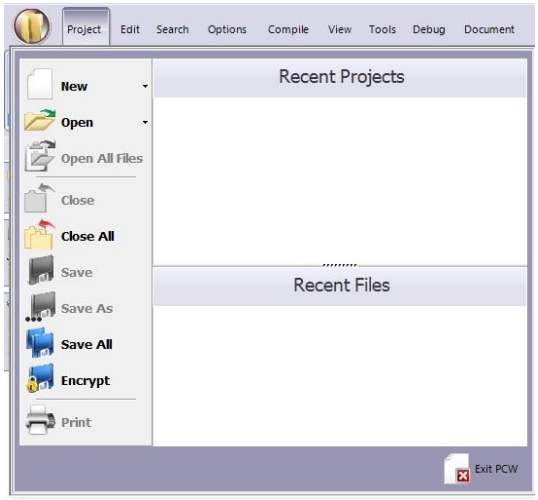
DERLEYİCİ C program "Başlat" menüsü altında "PIC C" adı altında yer alır. Programı başlatmak için sırasıyla "Başlat, Programlar, PIC C ve PIC C Compiler" yolu izlenir.

Program çalıştırıldığında arayüz ekranı açılır. Bu ekranın üstünde menü çubuğu ve menü içeriği araç çubuğunda yer alır. Menü çubuğundan seçilen "Project, Edit, Search, Options, Compile, View, Tools, Debug, Document ve User" menülerine ait içerikler menü içeriği araç çubuğundan da gösterilir.



Resim 1.7: DERLEYİCİ C programı arayüz ekranı

- **File menüsü:** Arayüz ekranının üst sol köşesinde bulunan daire içindeki klasör simgesine fare ile tıklanarak ulaşılır. Bu menü altında yer alan komutlar ve görevleri aşağıda kısaca verilmiştir.

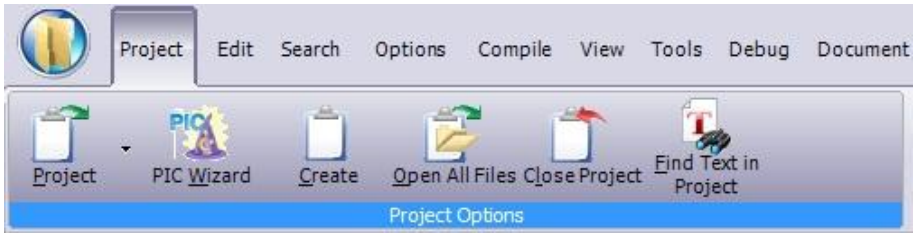


Resim 1.8: DERLEYİCİ C programı "File" menüsü

New : Yeni bir dosya açar.
Open : Önceden oluşturulmuş ve bilgisayarda kayıtlı olan dosyayı açar.
Close : Aktif olan (üzerinde çalışılan) dosyayı kapatır.

Close All : Açık olan tüm dosyaları kapatır.
Save : Dosyayı kaydeder.
Save As : Dosyayı farklı bir isimle ve/veya farklı bir dizine kaydeder.
Save All : Açık olan tüm dosyaları kaydeder.
Encrypt : Tanımlama dosyalarını şifreler.
Print : Dosyayı yazıcıya gönderir.
Recent Files : Kullanılan en son dosyaları listeler.
Exit : Programı kapatır.

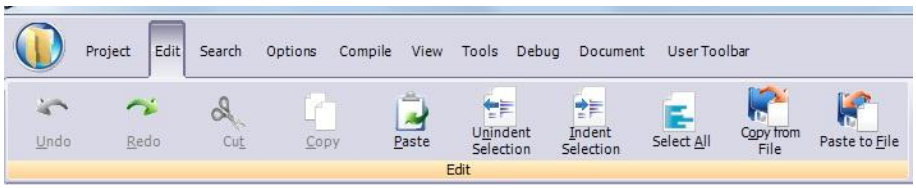
- **Project menüsü:** Project menüsü fare ile seçildiğinde menü içeriği araç çubuğunda aşağıdaki gibi görünür. Project menüsündeki komutların görevleri aşağıdaki gibidir.



Resim 1.9: DERLEYİCİ C programı "Project" menüsü

Project : Önceden kaydedilmiş Project dosyasını (pjt) açar.
PIC Wizard : Sihirbaz aracılığıyla (otomatik olarak) proje dosyası oluşturmayı sağlar.
Create : Manuel olarak proje dosyası oluşturmayı sağlar.
Open All Files : Proje dosyasına ait tüm dosyaları açar.
Close Project : Proje dosyasını açık tüm dosyaları ile birlikte kapatır.
Find Text In Project : İstenen metni tüm proje dosyaları içinde arar.

- **Edit menüsü:** Edit menüsü fare ile seçildiğinde menü içeriği araç çubuğunda aşağıdaki gibi görünür. Edit menüsündeki komutların görevleri aşağıdaki gibidir.

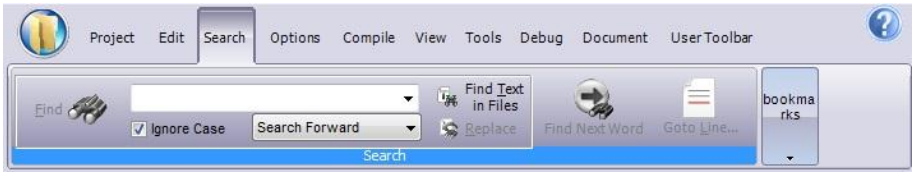


Resim 1.10: DERLEYİCİ C programı "Edit" menüsü

Undo : En son yapılan işlemi geri alır.
Redo : En son geri alınan işlemi eski haline getirir.
Cut : Seçili metni keser.
Copy : Seçili metni kopyalar.
Paste : Cut ve Copy işlemiyle hafızaya alınan metni istenen yere yapıştırır.

Unindent Selection : İmlecin bulunduğu satırdaki satır başıboşluğunu geri alır.
Indent Selection : İmlecin bulunduğu satıra satır boşluğu verir.
Select All : Tüm metni seçer.
Copy From File : Seçilen dosyadaki metni imlecin bulunduğu yere kopyalar.
Past To File : Seçilen metni istenen dosyaya yapıştırır.
Macros : Makro işlemleri yapar.

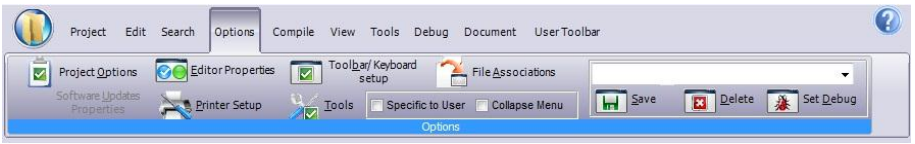
- **Search menüsü:** Search menüsü fare ile seçildiğinde menü içeriği araç çubuğunda aşağıdaki gibi görünür. Search menüsündeki komutların görevleri aşağıdaki gibidir.



Resim 1.11: DERLEYİCİ C programı "Search" menüsü

Find : İstenen metni açık olan dosyada arar.
Find Text in Project : İstenen metni tüm proje dosyalarında arar.
Goto Line : İstenen numaralı satıra gider.
Toggle Bookmark : Yer imi koyar.
Goto Bookmark : İmleç seçilen yer imine gider.

- **Options menüsü:** Options menüsü fare ile seçildiğinde menü içeriği araç çubuğunda aşağıdaki gibi görünür. Options menüsündeki komutların görevleri aşağıdaki gibidir.



Resim 1.12: DERLEYİCİ C programı "Options" menüsü

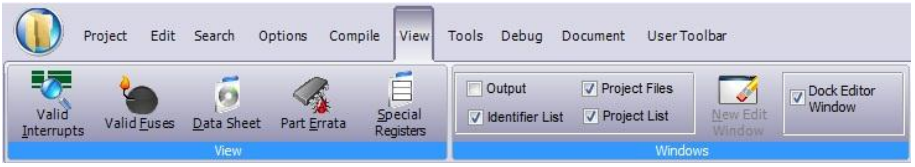
Project Options : Proje dosyasıyla ilgili ayarlamaları yapar.
Editor Properties : Derleyici programının ayarlarını yapar.
Tools : Menülerin gösterilmesini veya gizlenmesini ayarlar.
Software Updates Properties : Güncelleme hatırlatmasının hangi sıklıkta yapılacağını ayarlar.
Printer Setup : Yazıcı ayarlarını yapar.
Toolbar/Keyboard Setup : Araç çubuğu veya simgelerin eklenmesini veya çıkarılmasını ayarlar.

➢ **Compile menüsü:** Compile menüsü fare ile seçildiğinde menü içeriği araç çubuğunda aşağıdaki gibi görünür. Compile menüsündeki komutların görevleri aşağıdaki gibidir.

Resim 1.13: DERLEYİCİ C programı "Compile" menüsü

Compile : Projenin derlenerek "HEX" uzantılı dosyanın (denetleyiciye yüklenecek program) oluşmasını sağlar.
Build : Proje içindeki bir veya daha fazla dosyayı derler.
Lookup Part : Kullanılacak denetleyici modelini seçer.
Program Chip : Denetleyicinin programlama kartı ile programlanması komutu.
Debug : Hata ayıklamasının yapılmasını sağlar.
C/ASM List : C ile yazılmış programın Assembly karşılığını verir.
Symbol Map : Değişkenlerin kaydedildiği yeri ve derleme sonucu oluşturulan dosyaların yolunu gösterir.
Call Tree : Programda kullanılan fonksiyonları ve fonksiyon içindeki komutların bellekteki bilgilerini verir.
Statistics : Derlenen program ile ilgili bilgi verir.

➢ **View menüsü:** View menüsü fare ile seçildiğinde menü içeriği araç çubuğunda aşağıdaki gibi görünür. View menüsündeki komutların görevleri aşağıdaki gibidir.



Resim 1.14: DERLEYİCİ C programı "View" menüsü

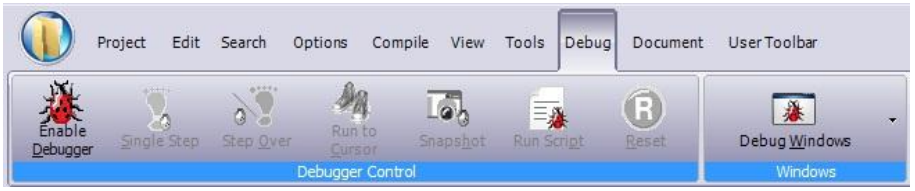
Valid Interrupts : Kullanılan denetleyicinin sahip olduğu kesmeleri gösterir.
Valid Fuses : Kullanılan denetleyiciye ait konfigürasyon ayarlarını yapar.
Data Sheet : Kullanılan denetleyiciye ait datasheet dosyasını açar.
Part Errata : Hata veri tabanını gösterir.
Special Registers : Denetleyicinin özel kaydedicilerini gösterir.
Output : Derleme sonucu oluşan hata ve uyarı bilgilerinin ekranın alt tarafında görünmesini sağlar.
New Edit Windows : Çalışılan dosyanın yeni pencerede gösterilmesini sağlar.
Dock Editor Window : Seçildiğinde editör penceresini ana pencereye sabitler.

➢ **Tools menüsü:** Tools menüsü fare ile seçildiğinde menü içeriği araç çubuğunda aşağıdaki gibi görünür. Tools menüsündeki komutların görevleri aşağıdaki gibidir.

Resim 1.15: DERLEYİCİ C programı "Tools" menüsü

Device Editor : Kullanılacak denetleyiciye ait özellikleri gösterir.
Device Selector : İstediğiniz özellikleri sunan denetleyiciye bulmayı sağlar.
File Compare : Seçilen iki dosyayı karşılaştırır.
Numeric Converter : Decimal, float, hex, unsigned ve signed türdeki değerlerin karşılığını verir.
Serial Port Monitor : Seri iletişim uygulamalarında Windows Hyper Terminal programıyla aynı işi yapan Siow. exe programını çalıştırır.
Disassembler : HEX uzantılı dosyadan ASM dosyasını üretir.
Convert Data to C : Binary, hex, integers ve floats dosyalarından #ROM ve CONST dosyalarına kod dönüşümü yapar.
Extract Cal Data : Hex dosyasındaki C başlık dosyasına giden ayar bilgilerini ayıklar.
Mach-X : Mach-X.exe programını çalıştırarak hex dosyasını denetleyiciye yükler (Uyumlu cihazın bağlı olması şartıyla).
ICD : ICD. exe programını çalıştırarak hex dosyasını denetleyiciye yükler (uyumlu cihazın bağlı olması şartıyla).

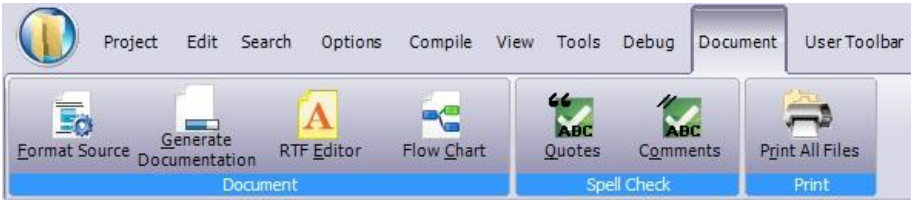
- **Debug menüsü:** Debug menüsü fare ile seçildiğinde menü içeriği araç çubuğunda aşağıdaki gibi görünür. Debug menüsündeki komutların görevleri aşağıdaki gibidir.



Resim 1.16: DERLEYİCİ C programı "Debug" menüsü

Enable Debugger : Hata ayıklama penceresini açar.
Single Step : Tek kaynak kodunu işleme sokar.
Step Over : Adım adım çalıştırır.
Run to Cursor : İmlecin bulunduğu koçu çalıştırır.
Snapshot : Hata ayıklama bilgilerinin kaydedilmesini sağlar.
Reset : Hata ayıklayıcıyı resetler.
Debug Windows : Hata ayıklama bilgilerinin görünmesini sağlar.

- **Documet menüsü:** Document menüsü fare ile seçildiğinde menü içeriği araç çubuğunda aşağıdaki gibi görünür. Document menüsündeki komutların görevleri aşağıdaki gibidir.



Resim 1.17: DERLEYİCİ C programı "Document" menüsü

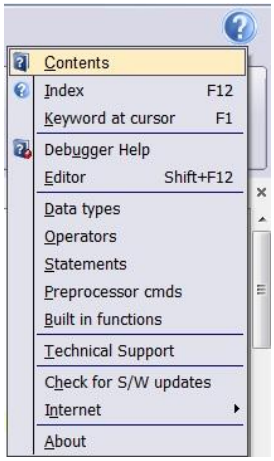
Format Source :Bazı karakter ve kelimelerin biçimlendirilmesini sağlar.
Generate Documendation: Derleme bilgilerini. rtf dosyası olarak oluşturur.
RTF Editor : RTF Editör programını çalıştırır.
Flow Chart : Akış diyagramı ve elektriksel devre çizmeyi sağlar.
Quotes : Tırnak içindeki sözcüklerin kontrolünü yapar.
Comments : Tüm açıklamaların sözcük kontrolünü yapar.
Print All Files : Projeye ait tüm dosyaları yazıcıya gönderir.

- **User Toolbar menüsü:** User Toolbar menüsü fare ile seçildiğinde menü içeriği araç çubuğunda aşağıdaki gibi görünür. User Toolbar menüsündeki komutların görevleri aşağıdaki gibidir.

Resim 1.18: DERLEYİCİ C programı "User Toolbar" menüsü

User… butonuna tıklayarak açılan pencere aracılığıyla kullanıcı istediği araçlardan oluşan kendine özel bir menü oluşturabilir. Açılan pencerenin sağ tarafından araç seçilerek "Add Item" butonu aracılığıyla aracın sol tarafa geçmesi sağlanır. Böylece kullanıcıya özel menü oluşturulmuş olur.

- **Help Menüsü:** Help menüsü fare ile tıklandığında içeriği açılır pencerede görünür. Help menüsündeki komutların görevleri aşağıdaki gibidir.



Resim 1.19: DERLEYİCİ C programı "Help" menüsü

Contents : Yardım dosyası içeriğini gösterir.
Index : Yardım dosyasındaki tüm dizinleri göstererek arama yapma olanağı sunar.
Keyword at Cursor : Kursör ile bir kod seçilerek bu komut çalıştırılırsa koda ait bilgileri gösterir.
Debugger Help : Hata ayıklamayla ilgili yardım bilgilerini gösterir.
Editor : Kısa yol tuşlarını gösterir.
Data types : Derleyicinin desteklediği veri türlerini gösterir.
Operators : Derleyicinin desteklediği C operatörlerini gösterir.
Statements : Derleyicide kullanılabilecek döngü fonksiyonlarını gösterir.
Preprocessor cmds : Derleyicide kullanılabilecek ön işlemci komutlarını gösterir.
Built in functions : Derleyicide olan fonksiyon listesini gösterir.
Technical Support : Destek almak için üreticiye e-posta gönderir.
Check for S/W updates : Programın güncel sürümü olup olmadığını kontrol eder.
Internet : DERLEYİCİ C'nin ana sitesine ve ilgili sitelere bağlantı sağlar.
About : Program hakkında bilgi verir.

1.2. DERLEYİCİ C Derleyicisi Temel Kavramları

1.2.1. DERLEYİCİ C Dilinde Kullanılan Operatörler ve Örnek Programlar

1.2.1.1. DERLEYİCİ C Veri Türleri

Diğer programlama dillerinde olduğu gibi C dilinde de kullanılan ifadelerin bir türü vardır ve bu veri türü kullanılan ifadenin içeriğini ve sınırlarını belirtir. Aşağıda verilen tabloda (Tablo 1.1) C dilindeki veri türleri ve DERLEYİCİ C'deki karşılıkları verilmiştir.

C Dili Veri Türü	DERLEYİCİ C Karşılığı	Açıklaması
short	int1	1 bitlik tamsayı
char	unsigned int8	Karakter
int	int8	8 bitlik tamsayı
long	int16	16 bitlik tamsayı
long long	int32	32 bitlik tamsayı
float	float32	32 bitlik ondalıklı sayı

Tablo 1.1: C dilindeki veri türlerinin DERLEYİCİ C derleyicisindeki karşılıkları

Veri türlerinin sınırları vardır. İyi bir program yazarı kullanılacak verilerin türlerini ve sınırlarını hesaba katmalı ve doğru tanımlamalar yapmalıdır. Aksi halde yapılan tanımlamaya bağlı olarak ya program hatalı çalışacak ya da gereksiz bellek kullanımı meydana gelecektir.

DERLEYİCİ C dilinde veri türlerinin önüne **signed** (işaretli) ya da **unsigned** (işaretsiz) tanımlamaları getirilerek veri sınırları belirlenir. Eğer herhangi bir tanımlama (signed/unsigned) kullanılmazsa DERLEYİCİ C derleyicisi veriyi **unsigned** olarak değerlendirecektir. Bu tanımlamalar "float" veri türü için kullanılmaz.

Aşağıda veri türlerinin **signed** (işaretli) ve **unsigned** (işaretsiz) durumları için değer aralıkları gösterilmiştir.

Veri Türü	Büyüklük	unsigned (işaretsiz)		signed (işaretli)	
		En Küçük Değeri	En Büyük Değeri	En Küçük Değeri	En Büyük Değeri
int1	1 bitlik tamsayı	0	1	Kullanılmaz	Kullanılmaz
int8	8 bitlik tamsayı	0	+255	-128	+127
int16	16 bitlik tamsayı	0	+65.535	-32.768	+32.767
int32	32 bitlik tamsayı	0	+4.294.967.295	-2.147.483.648	+2.147.483.647
float32	32 bitlik ondalıklı sayı	$-1{,}5x10^{45}$ ile $3{,}4x10^{38}$ arası			

Tablo 1.2: DERLEYİCİ C dilinde veri türlerinin sınırları

1.2.1.2. Operatörler ve Örnek Programlar

Sabit veya değişkenler üzerinde mantıksal, aritmetiksel vb. gibi işlemler yaparak yeni değerler üreten işaretlere operatör (işleç) denir. Aşağıda DERLEYİCİ C derleyicisi tarafından tanınan operatörler ve karşılıkları verilmiştir (Tablo 1.3).

Operatör	Açıklaması
=	Atama operatörü
+	Toplama operatörü
+=	Toplama işlemli atama operatörü (a+=1 ifadesi a=a+1 anlamındadır.)
++	Arttırma operatörü (a++ ifadesi a=a+1 anlamındadır.)
-	Çıkarma operatörü
-=	Çıkarma işlemli atama operatörü (a-=1 ifadesi a=a-1 anlamındadır.)
--	Azaltma operatörü (a -- ifadesi a=a-1 anlamındadır.)
*	Çarpma / Gösterici operatörü
*=	Çarparak atama operatörü (a*=2 ifadesi a=a*2 anlamındadır.)
&	VE / Adres operatörü
&=	VE işlemli atama operatörü (a&=b ifadesi a=a&b anlamındadır.)
^	ÖZEL VEYA operatörü
^=	ÖZEL VEYA işlemli atama operatörü (a^=b ifadesi a=a^b anlamındadır.)
|	VEYA operatörü
|=	VEYA işlemli atama operatörü (a|=b ifadesi a=a|b anlamındadır.)
/	Bölme operatörü
/=	Bölme işlemli atama operatörü (a/=2 ifadesi a=a/2 anlamındadır.)
==	Eşitlik operatörü
!=	Eşit değil operatörü
>	Büyüktür operatörü
>=	Büyük veya eşittir operatörü
<	Küçüktür operatörü
<=	Küçük veya eşittir operatörü
<<	Sola kaydırma operatörü
<<=	Sola kaydırma işlemli atama operatörü (a<<=b ifadesi a=a<<b anlamındadır.)
>>	Sağa kaydırma operatörü
>>=	Sağa kaydırma işlemli atama operatörü (a>>=b ifadesi a=a>>b anlamındadır.)
&&	Mantıksal VE operatörü (Mantıksal çarpma yapar.)
||	Mantıksal VEYA operatörü (Mantıksal toplama yapar.)
!	Mantıksal DEĞİL operatörü (Mantıksal tersleme yapar.)
~	Tümleyen alma operatörü
%	Modülüs operatörü (Bölümden kalanı verir.)
%=	Modülüs işlemli atama operatörü (a%=2 ifadesi a=a%2 anlamındadır.)
sizeof	Nesnenin veya türün bellekte kapladığı alanı veriri

Tablo 1.3: DERLEYİCİ C de kullanılan operatörler

Operatörler gerçekleştirdikleri işe göre aritmetiksel operatör, ilişkisel operatör, mantıksal operatör, bit operatörleri, *unary* operatörler ve özel operatörler olarak 6 grupta toplanırlar. Aşağıda bu operatör gruplarıyla ilgili örnekler verilmiştir. Böylece operatörlerin kullanımı ve çalışması daha kolay anlaşılacaktır.

Örnek 1: Aritmetiksel operatör kullanımı

a=5 ve b=2 değerlerini alan tamsayı türünde iki değişken için;
a+b işleminin sonucu 5+2 = **7** olur.
a-b işleminin sonucu 5-2 = **3** olur.
a*b işleminin sonucu 5*2 = **10** olur.
a/b işleminin sonucu 5/2 = **2** olur. Çünkü sonucun tamsayı türünde çıkması gerekir. ve 5'in 2'ye bölümünde bölüm 2 dir.
a%b işleminin sonucu 5%2 = **1** olur. Çünkü 5'in 2'ye bölümünde kalan 1'dir.

Örnek 2: İlişkisel (karşılaştırma) operatör kullanımı

a=5 ve b=2 için;
a>b işlemi 5, 2'den büyük olduğundan doğrudur ve TRUE (1) değeri üretilir.
a<b işlemi 5, 2'den küçük olmadığından yanlıştır ve FALSE (0) değeri üretir.
a==b işlemi 5, 2'ye eşit olmadığından yanlıştır ve FALSE (0) değeri üretir.
a!=b işlemi 5, 2'ye eşit olmadığından doğrudur ve TRUE (1) değerini üretir.

İlişkisel operatörlerde şart cümlesi doğru ise TRUE (1), yanlış ise FALSE (0) değeri üretilir. Üretilen bu TRUE ve FALSE değerleri "if-else" ve "while" kodlarıyla değerlendirilerek programa karar verme özelliği kazandırılır.

Örnek 3: Mantıksal operatör kullanımı

Mantıksal operatörler birden fazla şart cümlesini birleştirmek için kullanılırlar. Bir sayı mantıksal olarak yorumlandığında sayının değeri 0 ise FALSE (0) değerini, sayının değeri sıfırdan farklı ise TRUE (1) değerini üretir.

a=5&& -2 5 ve -2 sıfırdan farklı olduğundan her ikisi de TRUE (1) olarak kabul edilir. VE operatörü mantıksal çarpma yaptığından a=1*1 gibi işlem görür ve **a=1**olur.

b=5&&0 5 sıfırdan farklı olduğundan TRUE (1), 0 ise FALSE (0) olarak kabul edilir. VE operatörü mantıksal çarpma yaptığından b=1*0 gibi işlem görür ve **b=0** olur.

c=5||0 5 sıfırdan farklı olduğundan TRUE (1) ve 0 ise FALSE (0) olarak kabul edilir. VEYA operatörü mantıksal toplama yaptığından c=1+0 gibi işlem görür ve **c=1** olur.

d=0||-2 0 değeri FALSE (0) ve -2 sıfırdan farklı olduğundan TRUE (1) olarak kabul edilir. VEYA operatörü mantıksal toplama yaptığından d=0+1 gibi işlem görür ve **d=1** olur.

e=!5 5 sıfırdan farklı olduğundan TRUE (1) olarak kabul edilir. DEĞİL operatörü mantıksal tersleme yaptığından FALSE (0) olarak yorumlanır ve **e=0** olur.

Örnek 4: Bit operatör kullanımı

Bit operatörleri sayıyı bir bütün olarak değil bit düzeyinde değerlendirerek işlem yaparlar. Ayrıca bit operatörleri sadece tamsayı türündeki değişkenlerle kullanılabilirler.

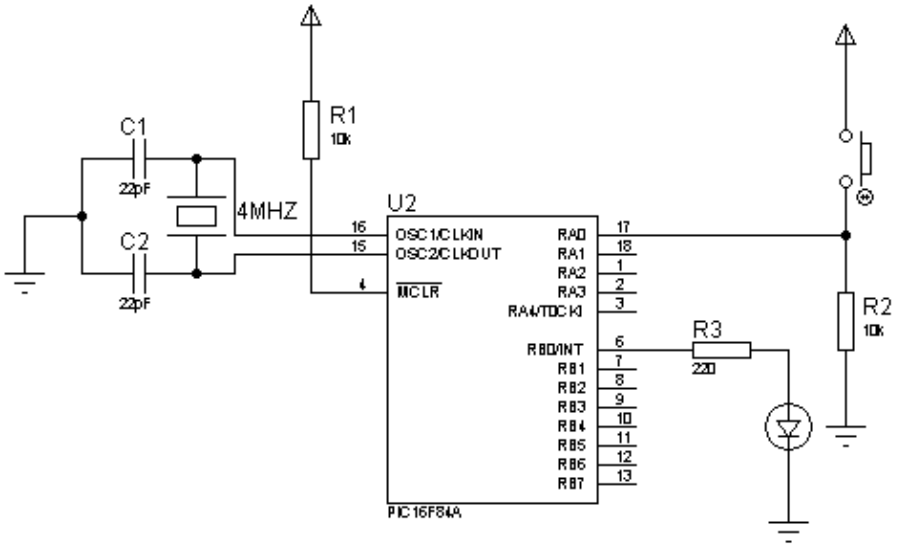
Bit VE işleminde karşılıklı her bit çarpılır, bit VEYA işleminde karşılıklı her bit toplanır, bit ÖZEL VEYA işleminde ise karşılıklı bitler aynı ise 0, farklı ise 1 değeri üretir.

a=0x23 a değişkeninin ikilik (binary) karşılığı a=0010 0011
b=0xB5 b değişkeninin ikilik (binary) karşılığı b=1011 0101

Bit VE (&)	Bit VEYA (|)	Bit ÖZEL VEYA (^)	Bit Öteleme
0010 0011 a	0010 0011 a	0010 0011 a	Kaybolur ← 0 010 0011 a
1011 0101 b	1011 0101 b	1011 0101 b	0100 011 0 ← 0 ile beslenir
0010 0001 a&b	1011 0111 a|b	1001 0110 a^b	0100 0110 c=a<<1

Bu bölümde DERLEYİCİ C dilinde operatörlerin kullanımıyla ilgili örnekler üzerinde durulacaktır. Verilen örnek kodların, programın tamamı olmadığını ve operatörlerin kullanımıyla ilgili bölümlerinden ibaret olduğunu unutmayın.

Örnek 5: Denetleyicinin, A0 pinine bağlanan butona basılma sayısını sayan ve sayı 10'a ulaştığında B0 pinine bağlı LED'i yakması istensin. Bu işlemi gerçekleştirecek program ve devre şeması aşağıdaki gibidir.



Şekil 1.1: Bir buton ile LED yakma devresi

Program Kodları:

```
#include "D:\ISIS\operator_kullanimi. h"
#use fast_io(a)
#use fast_io(b)
int basma_sayısı=0;
void main()
{
  setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);

set_tris_a(0x01);
set_tris_b(0x00);
  output_b(0x00);
  while(1)                    //Sonsuz döngü
{
    if(input(pin_a0)==1)      // A0 pinine tetikleme geldiğinde bir altındaki kodu çalıştırır.
        basma_sayisi++;       // basma_sayısı isimli değişkenin değerini 1 arttırır.

    if(basma_sayisi==10)      // basma_sayısı'nin değeri 10 ise bir altındaki kodu çalıştırır.
        output_high(pin_b0);  // B0 pinini lojik 1 (+5V) yaparak LED'i yakar.
}
}
```

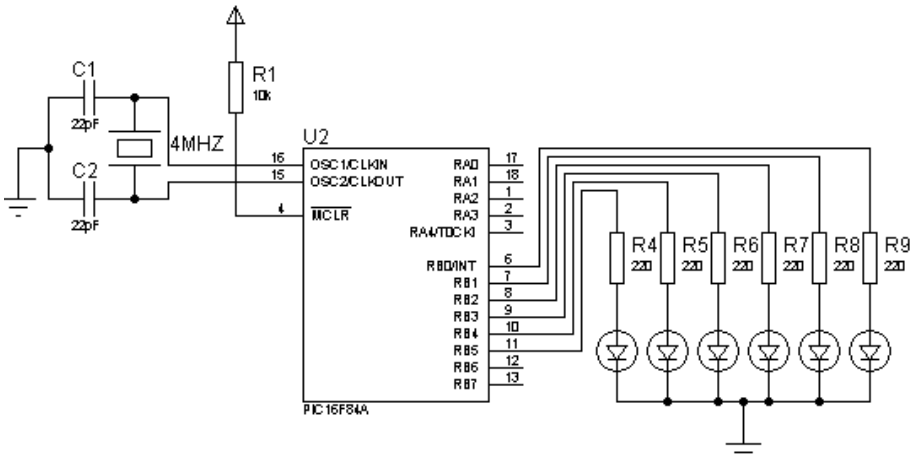
Yukarıdaki program ile unary operatörlerin (++) ve ilişkisel operatörlerin (==) kullanımı anlatılmıştır. Şimdilik programın "while(1)" ile başlayan bölümü üzerinde duracağız. Bu Programın çalışması kısaca şöyledir; "basma_sayısı" isimli bir değişken tanımlanmış ve atama operatörü ile (=) değeri 0 olarak ayarlanmıştır. Tetiklemenin ne zaman geleceği belli olmadığı için kodlar sonsuz döngü içine (while(1){ ….})yazılmıştır. Sonsuz döngü "while(1)" kodundaki parantez içerisine "1" değeri yazılarak sağlanmıştır. Çünkü "1" değeri sıfırdan farklı olduğundan mantıksal olarak parantez içerisi TRUE olur ve "while(TRUE)" koduyla aynı işlemi görür. Parantez içerisi TRUE olduğundan şart sağlanmış olarak değerlendirilecek ve sürekli bir döngü işlemi gerçekleşecektir.

Sonsuz döngü içindeki kodlar sırayla işlenmektedir. "if(input(pin_a0)==1)" kodu ile denetleyicinin A0 pinine tetikleme gelip gelmediği (tetikleme geldiğinde lojik 1 olur) kontrol edilmektedir. Bu koddaki "==" operatörü ile A0 pinindeki değer lojik 1'e eşitse "if" komutuna ait parantez içerisi TRUE olur ve bir altındaki kodun "basma_sayısı++" çalışmasını sağlar ve bu "basma_sayısı++;" kodu ile değişken değeri 1 artar.

Eğer tetikleme gelmemiş ise (A0 pinindeki değer lojik 1'e eşit değilse) "if" kodundaki parantez içerisi FALSE olur bir altındaki kod çalıştırılmadan program diğer koda "if(basma_sayısı==10)" atlar. Bu kod ile tetikleme işleminin 10 kez gerçekleşip gerçekleşmediği kontrol edilir. Eğer gerçekleşmiş ise (yani "basma_sayısı" isimli değişkeninin değeri 10 ise) bir altındaki kod olan "output_high(pin_b0);" kodu çalıştırarak B0 pini lojik 1 (+5V) seviyesine çekilir. Böylece B0 pinine bağlı led ışık verir.

Eğer “basma_sayısı” isimli değişkenin değeri henüz 10 olmamışsa buradaki “if” kodunun parantez içerisi şart sağlanmadığı için FALSE olur ve sonsuz döngü (while(1){…}) sayesinde program başa dönerek aynı işlemleri tekrar eder.

Örnek 6: Denetleyicinin, B portuna bağlanan 8 LED’in sırayla soldan sağa ve sağdan sola doğru sürekli olarak yandığı yürüyen ışık devresine ait programı yazalım. Program ve devre şeması aşağıdaki gibidir.



Şekil 1.2: Yürüyen ışık devresi

Program Kodları:

```
#include "C:\Users\PC\Desktop\ornek2.h"
#use fast_io(b) // B portu için port yönlendirmesi yapılıyor.
int a=1,i;    // a ve i adında int türünde değişkenler tanımlanmış ve a değişkenine 1 değeri
              //atanmış
void main()
{
  setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
set_tris_b(0x00);  // B portu çıkış olarak ayarlanıyor.
  output_b(0x00);   // B portu lojik 0 (GND) seviyesine çekilerek başlangıçta LED’lerin ışık
                    // vermesi engelleniyor.
  while(1)        // Sonsuz döngü
{
    for(i=0;i<7;i++)      // Döngü (her seferinde i nin değerini 1 arttırır ve i, 7’den küçük
                          // olduğu sürece çalışır)
{
      output_b(a);        // a nın değerini B portuna gönderir.
a=a<<1;                   // a nın değerini bit bazında bir sola öteler.
      delay_ms(250); // gecikme
}
    for(i=7;i>0;i--)      // Döngü (her seferinde i nin değerini 1 azaltır ve i, 0’dan büyük
                          // olduğu sürece çalışır.)
{
```

```
    output_b(a);        // a nın değerini B portuna gönderir.
a=a>>1;         // a nın değerini bit bazında bir sağa öteler.
    delay_ms(250); // gecikme
}
}
}
```

Yukarıdaki programda "#use fast_io(b);" komutuyla programda B portunun kullanılacağı belirtiliyor. "int a=1,i;" komutuyla a ve i adında tamsayı türünde iki değişken tanımlanarak a değişkeninin değeri 1 yapılıyor. Çünkü a değişkeni B portuna gönderilecek bilgiyi oluşturmakta ve sağdan başlayarak her seferinde sadece bir LED yanacağından başlangıçta B0 (RB0) pinin lojik 1 diğer pinlerin lojik 0 seviyesinde olması gerekmektedir. Tamsayı türündeki 1'in binary karşılığı 0b00000001'dir ve bu bilgi B portuna gönderildiğinde sadece B0 (RB0) pini lojik 1 olur.

"set_tris_b(0x00);" komutuyla B portunun çıkış olarak kullanılacağı belirtiliyor. "output_b(0x00);" komutu ise devreye enerji verildiğinde led'lerin ışık vermemesi için B portunu lojik 0 (GND) seviyesine çeker.

Devrenin sürekli olarak çalışması için "while(1) {…}" komutu kullanılmış ve diğer kodlar bu koda ait parantezler içerisine yazılmıştır. Böylece "while" komutuna ait "{…}" parantezler içerisine yazılan kodların devrede enerji olduğu sürece çalışması sağlanmıştır.

LED'lerin önce sola sonra sağa doğru sırayla yanmasını sağlamak için iki adet "for" döngüsü kullanılmıştır. "for(i=0;i<7;i++)" komutu i değişkeninin değerini 0'dan başlatarak her döngüde değerini 1 arttırır (i++ ile) ve döngü, i'nin değeri 7'den küçük olduğu sürece devam eder. Her döngüde "for" komutundan sonra bulunan "{..}" parantezler içerisindeki kodlar çalıştırılır. Toplam 8 adet LED'imiz bulunduğundan en soldaki (RB7 ye bağlı) LED'in yanması için toplam 8 adımlık döngü gerekmektedir. i değişkeni 0 değerinden başladığında i=6 olduğunda 7 adım tamamlanmış olur ve hala i'in değeri (6), 7 den küçük olduğundan döngü bir adım daha devam eder ve böylece 8 adımlık döngü tamamlanmış olur. Bu sebepten dolayı "i<7" yazılmıştır.

"output_b(a);" komutu ile a değişkeninin değeri B portuna gönderilir. "a=a<<1;" komutu ile a değişkeninin değeri bit bazında 1 bit sola ötelenir en soldaki bit (MSB biti) kaybolur ve en sağa (LSB biti) 0 eklenir. Döngü boyunca 0b00000001 olan a değişkeninin değeri "a=a<<1;" komutu sayesinde sıra ile 0b00000010, 0b00000100, … ,0b10000000 olana kadar değişir. "delay_ms(250);" komutu ile LED'lerdeki değişimi rahat görebilmek için 250 ms'lik gecikme verilmiştir. *a* değişkeninin değeri 0b10000000 olduğunda yani 8 adım sonunda ilk "for" döngüsü tamamlanarak ikinci "for" döngüsüne geçilir.

"for(i=7;i>0;i--)" ifadesi ilk "for" komutu gibi çalışır. Tek farkı burada i değişkeninin başlangıç değeri 7 olarak ayarlanmış ve değeri sıfırdan büyük olduğu sürece bir azaltılmaktadır. Bu ifade yerine "for(i=0;i<7;i++)" ifadesi yazılsaydı yine program sorunsuz çalışırdı çünkü amacımız işlemi 8 kez tekrar etmek. Ancak bu şekilde de yazılabileceğini görmeniz açısından bu yazım tercih edilmiştir. Bu döngü içindeki kodlara bakıldığında tek farkın "a=a>>1;" ifadesi olduğu görülmektedir. Bu ifade ile a değişkeninin değeri (bir önceki döngüde aldığı son değer) binary olarak 0b10000000'dan başlanarak her döngü tekrarında

sırayla 0b01000000, 0b00100000,…, 0b00000001 bit bazında bir bit sağa öteleme yapmaktır. Sağa ötelemede görüldüğü üzere her seferinde en soldaki bit (LSB) kaybolmakta ve en sağa (MSB) 0 değeri eklenmektedir. Böylece ışık veren LED her seferinde soldan sağa doğru sırayla değişmektedir (a değişkeninin binary karşılığındaki 1 değeri). 8 adım sonunda ikinci "for" döngüsü de tamamlanarak sonsuz döngü "while(1)" nedeniyle bu işlemler baştan itibaren tekrar eder.

1.3. DERLEYİCİ C ile Program Derlemek ve Denemek

1.3.1. Diziler

Aynı veri türünde ve aynı isim altında birden fazla veriyi tutmak için kullanılan yapılara dizi denir. Dizilerde tutulan elemanlar hafızaya bir biri ardına yerleştirilir. Bu özelliğinden dolayı dizi elemanına indis numarası sayesinde sırayla ya da rastgele erişilebilir. Dizilerin bu özelliği bazı uygulamalarda programcıya büyük kolaylık sağlar.

Dizi tanımlaması **<veri türü><dizi adı><[eleman sayısı]>** şeklinde yapılır;

```
char a[5];
char isim[5]={'K', 'İ', 'T', 'A', 'P'};
char isim[ ]={"KİTAP"};
int sayi[4];
int rakam[ ]={0,1,2,3,4,5,6,7,8,9};
```

Yukarıda verilen dizi tanımlamalarının hepsi doğrudur. Görüldüğü üzere tanımlama yapılırken değer ataması da yapılabilir. Tanımlama sırasında değer ataması yapılacaksa eleman sayısı isteğe bağlı olarak yazılmayabilir. Bu durumda derleyici eleman sayısını kendisi hesaplar.

Dizilerde ilk elemanına indis değeri 0'dır. Dolayısıyla son elemana ulaşmak için eleman sayısının bir eksiği indis numarası olarak kullanılmalıdır.

Aşağıda 7 elemanlı "int sayi[ ]={3,1,42,38,14,125,16};" dizisi için örnekler verilmiştir.

```
a=sayi[0];      // a=3 olur. Çünkü indis olarak "0" yazılmış ve dizinin ilk elemanı 3'tür.
a=sayi[1];      // a=1 olur. Çünkü indis olarak "1" yazılmıştır.
a=sayi[4];      // a= 14 olur.
a=sayi[6];      // a=16 olur.
```

Dizilerde indis numarası yerine tamsayı (int) türünde olmak şartıyla değişken de kullanılabilir. Aşağıda bu kullanıma örnek verilmiştir.

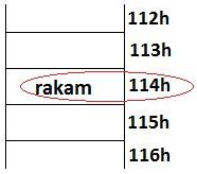
Int türündeki i değişkeni ve char türündeki a değişkeni için char x[5]={'A', 'h', 'm', 'e', 't'} dizisinde i değişkeni indis olarak kullanıldığında;

i=0 için a=x[i]; ifadesinde a değişkeninin değeri ‘A’ karakteri olur.
i=2 için a=x[i]; ifadesinde a değişkeninin değeri ‘m’ karakteri olur.
i=4 için a=x[i] ifadesinde a değişkenin değeri ‘t’ karakteri olur.

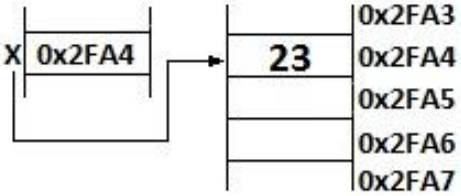
1.3.2. İşaretçiler (Pointer)

Pointer (gösterici) temel olarak bir nesnenin (değişken, sabit vb.) RAM bellekteki adresini gösteren veri tipidir. Göstericiler doğrudan bellek adresinde erişim sağladıklarından programların daha hızlı ve verimli olmasını sağlarlar.

Program içerisinde “int rakam;” şeklinde bir değişken tanımlaması yapıldığında derleyici tarafından bu değişken RAM bellekte bir adrese yerleştirilir (Şekil 1.3). Değişkenin bellekte yerleştirildiği adres verisini tutan nesnelere pointer (gösterici) denir.



Şekil 1.3: Değişkenin RAM bellekte tutulması



Şekil 1.4: Pointer kullanımı

Göstericiler; <tür ismi>< * ><gösterici ismi> şeklinde bildirilirler. Buradaki “tür ismi” göstericinin tuttuğu adreste bulunan veri türünü ifade eder. Yani gösterici char türünde bir değişkenin adresini tutuyorsa <tür ismi> alanına “char”, float türünde bir değişkenin adresini tutuyorsa “float” yazılmalıdır.

int *a; // a adında gösterici tanımlanıyor ve göstericinin tuttuğu adreste int türünde veri saklanıyor.
char *b; // b adında gösterici tanımlanıyor ve göstericinin tuttuğu adreste char türünde veri saklanıyor.
float *c; // c adında gösterici tanımlanıyor ve göstericinin tuttuğu adreste float türünde veri saklanıyor.

Örnek:

int *x;

x=(int*) 0x2FA4;

*x=23;

Yukarıdaki örnekte; “int *x;” ifadesiyle int türünde ve “x” adında bir gösterici tanımlanmıştır. “x=(int *) 0x2FA4;” ifadesiyle bu göstericiye hex olarak 2FA4 adresi atanmıştır. “*x=23;” ifadesiyle de göstericinin tuttuğu adrese 23 değeri atanmıştır.

1.3.3. Yapılar ve Fonksiyonlarla Program Yazma

Yapılar, kullanıcıya kendi veri türünü oluşturma imkânı verir. Oluşturulan bu yeni veri türü aynı ya da farklı veri türlerini içinde barındırabilir. Örneğin; bir sınıftaki öğrencilerin adını, numarasını ve notlarını girebileceğimiz bir program yapmak isteyelim. Yapıları kullanmadan bu programı iki şekilde yapabiliriz. Bunlardan birincisi her bir öğrenci için ayrı ayrı adını, numarasını ve notlarını tutacak değişkenler tanımlamak, ikincisi ise bu iş için dizileri kullanmaktır. Ancak bu iki yöntem de hem pratik hem de değişkenler hafızada yer işgal edeceğinden kullanışlı değildir. Böyle durumlarda yapılar yardımımıza koşar. Yapı bildirimi şu şekildedir;

```
struct <yapı ismi>
{
  <veri türü><değişken_1>;
  <veri türü><değişken_2>;
  …
  };
```

```
struct başarı_durumu
{
        char  ad[20];
        char soyad[20];
        int ogr_no;
        int not_1;
        int not_2;
        int not_3;
};
```

Yukarıda yapılan işlem ile sadece yapı bildirimi yapıldı. Şimdi yapı değişkeni tanımlanmalıdır. Yapı değişkeni;

struct <yapı ismi><değişken ismi>; **struct başarı_durumu öğrenci;**

Yukarıdaki bildirim ile öğrenci adında yeni bir yapı değişkenimiz oluşturulmuştur. Bu değişkenin veri türü başarı_durumu dur. Yapı elemanına değer atamak için;

```
strcpy(ogrenci. ad, “Osman”);
strcpy(ogrenci.soyad, “GÖNÜLAL”);
ogrenci.ogr_no=45;
ogrenci.not_1=60;
ogrenci.not_2=55;
ogrenci.not_3=80;
```

şeklinde olur.

Dikkat ederseniz char türündeki elemanlar ile int türündeki elemanlara değer atama farklıdır.

1.3.4. 16F ve 18F Serisi İşlemcilerle Program Yazılımı

PIC işlemci serileri değişik OPCODE uzunluklarına sahiptirler. Örneğin PIC12F675 14-Bit Opcode uzunluğuna, PIC16C57 12-Bit Opcode uzunluğuna, PIC18F452 16-Bit Opcode uzunluğuna sahiptir. Bu nedenle DERLEYİCİ C derleyicisi değişik Opcode uzunluğundaki işlemciler için değişik derleyiciler içermektedir. Bunlar PCB, PCM, PCH, ve PCD derleyicileridir. Kullanacağınız denetleyiciye bağlı olarak derleyicinizin doğru birimi barındırdığından emin olunuz.

PCB : 12-Bit Opcode, Baseline
PCM : 14-Bit Opcode, Midrange
PCH : 16-Bit Opcode, PIC18
PCD :16-Bit işlemci, PIC24/dsPIC (PIC24 ve dsPIC serisi 16-Bit, diğer seriler 8-bit denetleyicidir.)

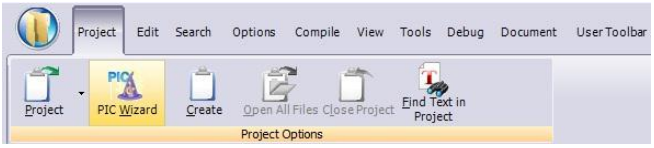
1.3.5. Derleyici Kullanımı

DERLEYİCİ C derleyicisinde program yazmaya başlamak için öncelikle proje dosyası oluşturulmalıdır. Proje dosyası, sihirbaz kullanarak (otomatik) veya manuel (el ile) olmak üzere iki şekilde oluşturulabilir.

1.3.5.1. Sihirbaz Kullanarak Proje Oluşturma

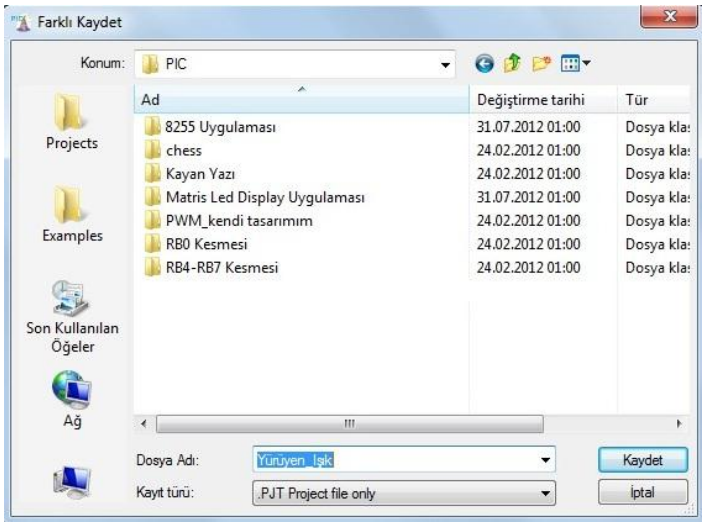
Sihirbaz kullanarak proje dosyası oluşturulduğunda DERLEYİCİ C derleyicisi, mikrodenetleyiciye ait (portların giriş/çıkış olarak yönlendirilmesi, kullanılacak denetleyicinin seçimi, Timer, İnterrupts vb.) birçok ayarlamayı sihirbaz aracılığıyla kod yazmadan yapma imkânı sunar.

Yeni bir proje dosyası oluşturmak için PIC C programında "Project" menüsüne tıklandığında araç çubuğunda görünen "PIC Wizard" komutu seçilir (Resim 1.20).



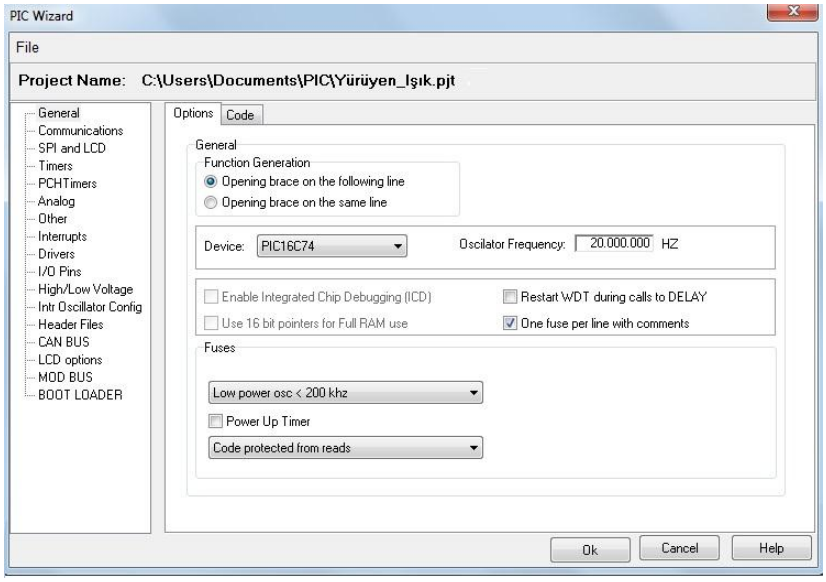
Resim 1.20: Project menüsü ve PIC Wizard

Açılan "**Farklı Kaydet**" penceresinden proje dosyasının kaydedileceği konum belirlenerek, projeye isim verilir ve kaydet butonuna basılır (Resim 1.21).



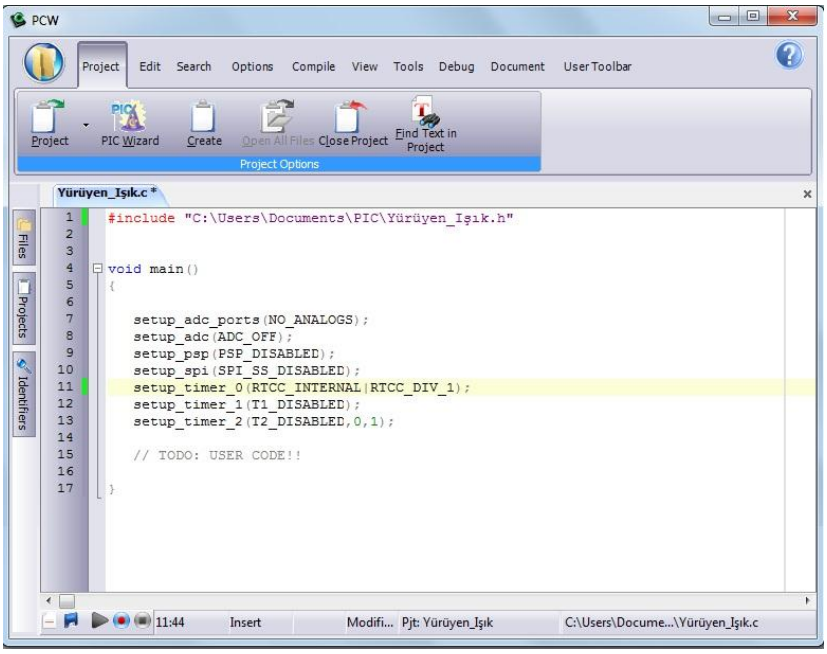
Resim 1.21: DERLEYİCİ C Derleyicisi "Farklı Kaydet" penceresi

Karşınıza PIC Wizard penceresi gelir. Bu pencere aracılığıyla kullanılacak denetleyici türü, denetleyici osilatör frekansı, portların giriş/çıkış ayarları, projeye eklenecek driver dosyaları ve kullanacağınız kesmeleri gibi birçok ayarlama yapabilirsiniz (Resim 1.22).



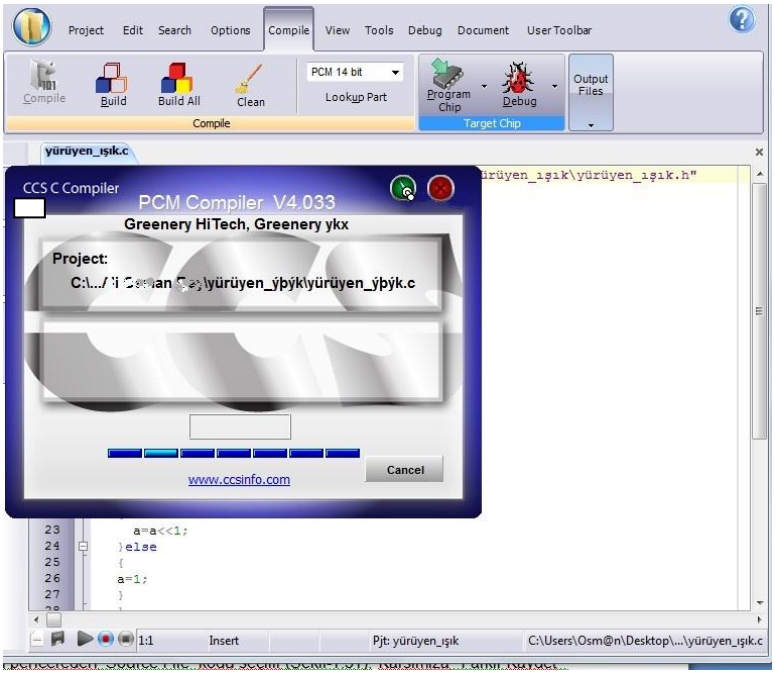
Resim 1.22: DERLEYİCİ C Derleyicisi "PIC Wizard" penceresi

PIC Wizard penceresinde gerekli ayarlamaları yapıp OK butonuna bastığınızda editör penceresi açılır (Resim 1.23). Artık programınızı yazmaya başlayabilirsiniz.



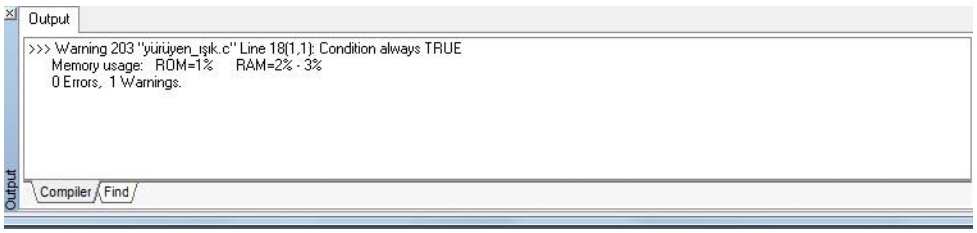
Resim 1.23: Otomatik oluşturulan proje dosyası

Program yazımı bittikten sonra sıra derleme işlemine gelir. Programı derlemek için "Compile" menüsü seçilerek araç çubuğunda bulunan "Compile" butonu tıklanır (Resim 1.24).



Resim 1.24: Programın derlenmesi

Derleme işleminin başarılı olup olmadığı, eğer programda hata varsa, hatanın hangi satırda olduğu DERLEYİCİ C derleyici programının alt tarafında bulunan "Output penceresi" aracılığıyla gösterilir (Resim 1.25).

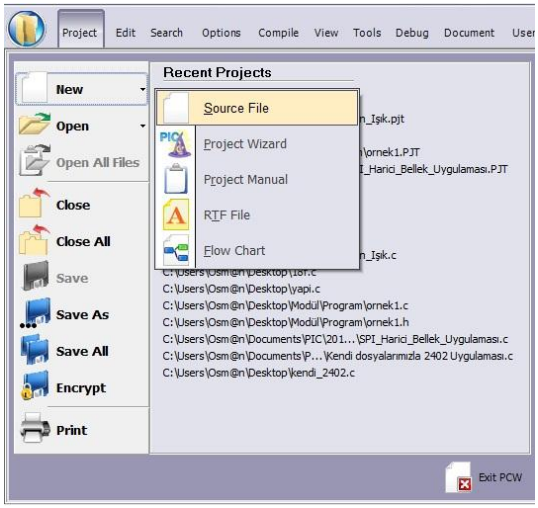


Resim 1.25: Output penceresi

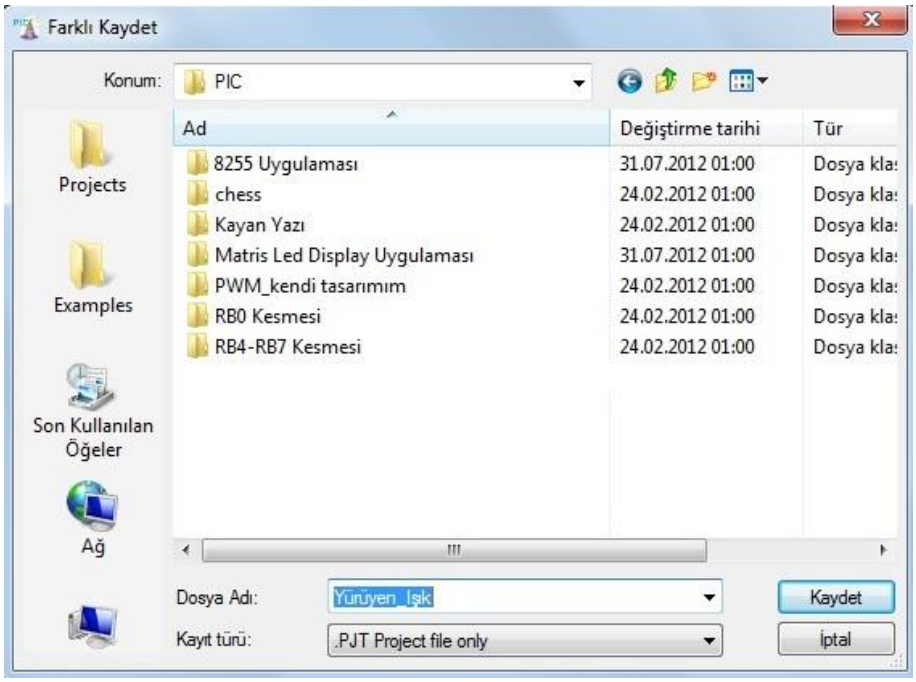
Eğer bu pencerede "0 Errors" yazısı varsa programın derlenmesi başarıyla gerçekleştirilmiştir.

1.3.5.2. Manuel Olarak Proje Dosyası Oluşturma

Manuel olarak proje oluşturmak için öncelikle File menüsü altında yer alan NEW komutu seçilir. Açılan pencereden "Source File" kodu seçilir (Resim 1.26). Karşımıza "Farklı Kaydet" penceresi gelir. Bu pencerede proje dosyasının oluşturulacağı konum ve proje ismi belirlenerek "Kaydet" butonuna basılır.

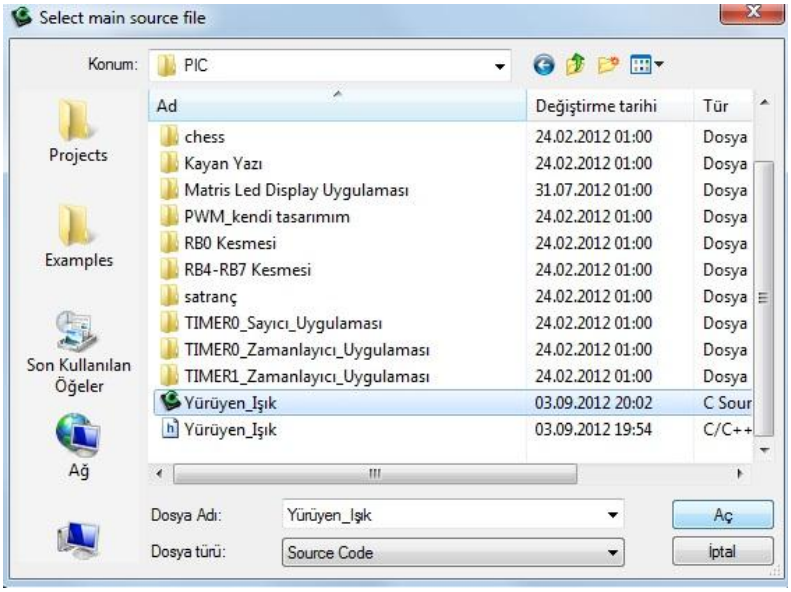


Resim 1.26: Manuel proje oluşturma



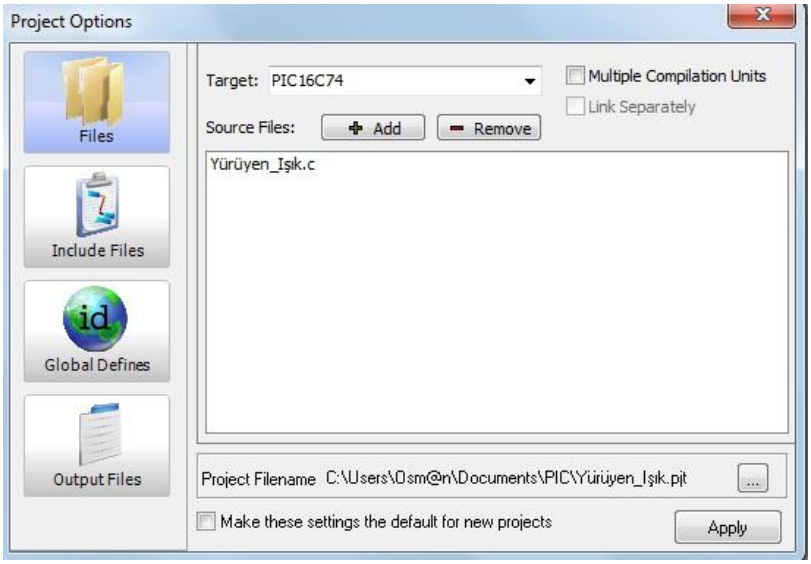
Resim 1.27: Farklı kaydet penceresi

Daha sonra Project menüsünden Create ya da File menüsünden Project Manuel komutları çalıştırılır. Karşımıza kaynak dosya seçim (Select main source file) penceresi gelir (Resim 1.28). Bu pencereden oluşturulan kaynak dosya seçilir ve "Aç" butonuna basılır.



Resim 1.28: Kaynak dosya seçim penceresi

Daha sonra karşımıza “Project Options” penceresi gelir (Resim 1.29). Buradan kullanılacak denetleyici seçilerek “Apply” butonuna basılır.

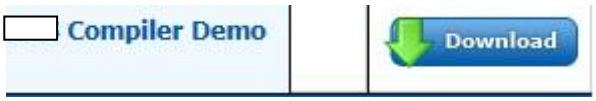
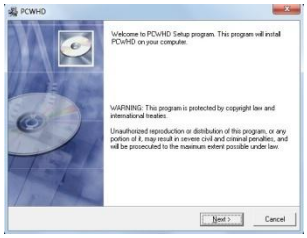


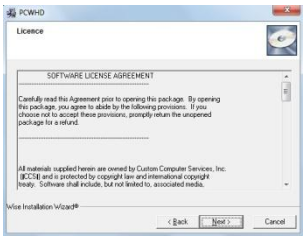
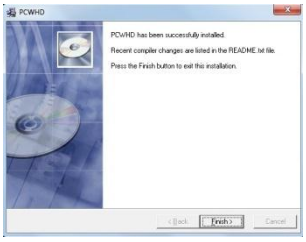
Resim 1.29: Project Options penceresi

Program kodları yazılır ve derleme işlemi için “Compile” menüsü seçilerek, araç çubuğunda bulunan “Compile” butonu tıklanarak derleme işlemi gerçekleştirilir.

UYGULAMA FAALİYETİ

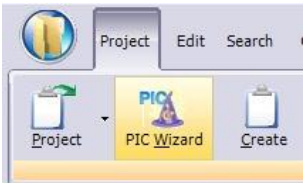
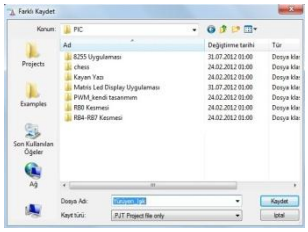
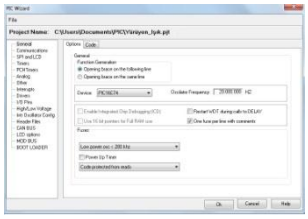
DERLEYİCİ C derleyicisini bilgisayara kurunuz.

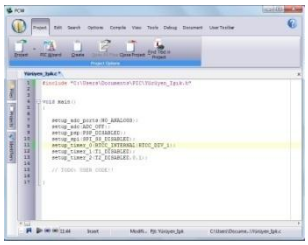
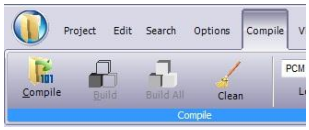
İşlem Basamakları	Öneriler
➢ DERLEYİCİ Compiler Demo yazısının karşısında bulunan "Download" butonuna tıklayınız.  	➢ Bilgisayarınızın güvenliği açısından programı sadece verilen internet sitesinden indirin.
➢ Demo sürüm için gerekli olan formu doldurarak "Submit" butonuna basınız. 	➢ Forma kural dışı bilgi girişi yapıldığında bir sonraki aşamaya geçmeyecek ve hatalı olan yerler formun üzerinde kırmızı renkte gösterilir.
➢ İsim ve e-posta formunu doldurarak "Download Demo" butonuna basınız. 	
➢ Karşınıza gelen pencereden setup programının bilgisayarınızda saklanacağı dizini seçiniz ve kaydet butonuna basınız.	
➢ Yükleme tamamlandığında kaydettiğiniz dizini açınız ve setup dosyası üzerine fare ile çift tıklayınız.	
➢ Karşınıza "Kurulum açılış ekranı" gelecek. NEXT butonuna basınız.  	

➢ Karşınıza “Lisans sözleşme ekranı” gelecek. NEXT butonuna basınız.  	
➢ Dizin seçme ekranından programın kurulmasını istediğiniz dizini seçerek NEXT butonuna basınız. 	
➢ Kurulum başlama ekranı geldiğinde NEXT butonuna basarak kurulum işlemini başlatınız. 	
➢ Dosyalar bilgisayarınıza kurulacak ve son olarak kurulum bitiş ekranı karşınıza gelecektir. Bu ekran üzerinde bulunan FINISH butonuna basarak kurulumu sonlandırınız.  	

UYGULAMA FAALİYETİ

Otomatik proje dosyası oluşturunuz ve projeyi derleyiniz.

İşlem Basamakları	Öneriler
➢ DERLEYİCİ C programını başlatınız.	➢ Programı masa üzerine atılan PIC C kısa yoluna çift tıklayarak ya da Başlat, Programlar, PIC C yolunu izleyerek PIC C Compiler komutunu fare ile seçerek başlatabilirsiniz.
➢ Program başladığında "Project" menüsünü fare ile seçiniz ve menü çubuğundan "PIC Wizard" komutunu çalıştırınız. 	
➢ Karşınıza "Farklı Kaydet" penceresi gelecek. Bu pencereden oluşturacağınız projenin konumunu ve adını belirterek "Kaydet" butonuna basınız. 	➢ DERLEYİCİ C de bir proje oluşturulduğunda ve derlendiğinde çok sayıda dosya oluşacağından her projeyi ayrı klasörlerde saklamanız daha iyi olur.
➢ Karşınıza PIC Wizard penceresi gelecek. Bu pencere aracılığıyla kullanacağınız denetleyiciyi (PIC 16F877A), denetleyici frekansını (4000000) ve Fuses alanından Crystal osc<=4MHZ seçerek OK butonuna basınız. 	➢ PIC Wizard penceresinden daha fazla ayarlama yapılabilir. Ancak bizim için şimdilik bu ayarların yapılması yeterli olacaktır.

➢ Proje dosyası oluşturularak karşımıza projeyle ilgili kodlar gelir. 	
➢ Proje dosyasına bölüm 1.2.1.2 de verilen “Örnek 5”e ait kodları yazınız.	
➢ “Compile” menüsünü fare ile seçerek menü çubuğundan	
➢ “Compile” komutunu çalıştırınız. 	
➢ Derleme işlemi yapılarak, .hex dosyası oluşturulmuştur.	➢ Derleme işlemi sonunda ekranın alt bölümünde “output” penceresi görünür. Bu pencerede “Error” yazısının başında 0 değeri yazılı ise derleme işlemi başarılı olmuştur. Aksi halde işlem başarısız olmuştur.

KONTROL LİSTESİ

Bu faaliyetler kapsamında aşağıda listelenen davranışlardan kazandığınız becerileri Evet, kazanamadığınız becerileri Hayır kutucuğuna (X) işareti koyarak kendinizi değerlendiriniz.

Değerlendirme Ölçütleri	Evet	Hayır
1. İnternetten bilgisayarınıza dosya indirebiliyor musunuz?		
2. Bilgisayarınıza program kurabiliyor musunuz?		
3. Bilgisayara kurulu programı çalıştırabiliyor musunuz?		
4. Program menüleri hakkında fikir sahibi oldunuz mu?		
5. Sihirbaz kullanarak yeni bir proje oluşturabildiniz mi?		
6. Proje dosyası üzerinde programı yazabildiniz mi?		
7. Programı derleyebildiniz mi?		

DEĞERLENDİRME

Değerlendirme sonunda "Hayır" şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınız "Evet" ise "Ölçme ve Değerlendirme"ye geçiniz.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki cümleleri dikkatlice okuyarak boş bırakılan yerlere doğru sözcüğü yazınız.

1. C programlama dili hem …….. kadar alt düzeye inebilen hem de ……yakın olması nedeniyle kolay öğrenilebilen bir dildir.

2. ……. ANSI C'nin temel tanımlamalarının büyük çoğunluğunu kabul eder ve diğer C temelli mikrodenetleyici derleyicilerine göre C dili ile daha fazla uyum içindedir.

3. Önceden oluşturulmuş ve bilgisayarda kayıtlı olan dosyayı açmak için …….. menüsü altında yer alan ……..komutu kullanılır.

4. Otomatik proje dosyası oluşturmak için …….. menüsü altında yer alan …….komutu kullanılır.

5. Manuel olarak proje dosyası oluşturmak için ……. menüsü altında yer alan ……komutu kullanılır.

6. C ile yazılmış programın assembly karşılığına ……. menüsü altında yer alan …….komutuyla ulaşılır.

7. Yazılan program; ……. menüsü altında yer alan …….komutu ile derlenir.

8. Yazılan programa akış diyagramı ve elektriksel devre oluşturmak için ……. menüsü altında yer alan …….komutu kullanılır.

9. "signed *int8 a;"* şeklinde tanımlanan a değişkeninin alabileceği değer aralığı ……..ile ……..arasındadır.

10. Tamsayı türünde tanımlanan bir değişkene 7/2 işleminin sonucu atıldığında (a=7/2;) değişkenin değeri ……..olur.

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ–2

AMAÇ

DERLEYİCİ C ile basit uygulamalar yapabileceksiniz.

ARAŞTIRMA

- Piyasada en çok tercih edilen denetleyicilerin neler olduğunu araştırınız.
- Denetleyici seçiminde nelere dikkat edildiğini araştırınız.
- Günlük hayatımızda yer eden hangi cihazlarda denetleyicilerin kullanıldığını araştırınız.

Bu araştırmaları yapmak için çevrenizde ya da internet üzerinde elektronik malzeme satışı yapan ya da mikro denetleyici programcılığıyla uğraşan kişilerden ve mikrodenetleyiciler üzerine yazılan kitaplardan faydalanabilirsiniz.

2. DERLEYİCİ C İLE UYGULAMALARI

2.1. Port Giriş/Çıkış İşlemleri ve Örnek Program Yazılımları

2.1.1. SET_TRIS_X( ) Komutu

Bu komut kullanılacak denetleyiciye ait portların ve bu portlara ait pinlerin giriş/çıkış olarak ayarlanmalarını sağlar. Komutta yer alan “X” yerine port ismi ve parantez içerisine “( )” 8 bitlik hex türünde değer girilir. Çıkış olarak yönlendirilecek pin için “0”, giriş olarak yönlendirilecek pin için ise “1” değeri kullanılır. LSB (en soldaki) biti X0 pini, MSB (en sağdaki) bit ise X7 pinidir. Komutun kullanımı aşağıda örneklerle birlikte verilmiştir.

set_tris_X(değer);

set_tris_a(0x00); // 0x00 sayısının binary karşılığı 0b00000000'dir. Bu nedenle A portunun tüm pinlerin çıkış olarak yönlendirilmiştir.

set_tris_b(0xFF); // 0xFF sayısının binary karşılığı 0b11111111'dir. Bu nedenle B portunun tüm pinlerin giriş olarak yönlendirilmiştir.

set_tris_c(0x25); // 0x25 sayısının binary karşılığı 0b00100101'dir. Bu nedenle C portunun C0, C2 ve C5 pinleri giriş, diğer pinlerin ise çıkış olarak yönlendirilmiştir.

2.1.2. GET_TRIS_X( ) Komutu

Bu komut "set_tris_X (deger)" komutu ile porta atanan değer ile geri döner. Yani Tris kaydedici değerini okur. Bu komutta "X" yerine port ismi yazılır. Komutun kullanımı; **deger=get_tris_a();** şeklindedir.

set_tris_b(0x0F);
deger=get_tris_b(); // değer değişkeninin değeri 0x0F olur.

2.1.3. OUTPUT_X( ) Komutu

Bu komut ile bir portun tüm pinlerine bilgi gönderilir. Gönderilen değerin binary karşılığında "1" bilgisi giden pin lojik-1 (+5V) seviyesine, "0" bilgisi giden pin lojik-0 (GND) seviyesine çekilir. Komutun kullanımı;

output_a(deger); şeklindedir. "deger" yerine int, hex ya da binary türden sayılar ile değişkenler kullanılabilir.

output_a(0x0F); // A portunun A0, A1, A2 ve A3 pinlerine lojik-1 bilgisi, diğer pinlerine lojik-0 bilgisi gönderir.

2.1.4. OUTPUT_HIGH( ) Komutu

Bu komut ile istenen portun istenen pini lojik-1 (+5V) seviyesine çekilir. Parantez içerisine pin ismi yazılır. Kullanımı;

output_high(pin ismi); şeklindedir.

output_high(pin_a2); // A portunun A2 pini lojik-1 yapılır.
output_high(pin_c5); // C portunun C5 pini lojik-1 yapılır.

2.1.5. OUTPUT_LOW( ) Komutu

Bu komut ile istenen portun istenen pini lojik-0 (GND) seviyesine çekilir. Parantez içerisine pin ismi yazılır. Kullanımı;

output_low(pin ismi); şeklindedir.

output_low(pin_a0); // A portunun A0 pini lojik-0 yapılır.
output_low(pin_b3); // B portunun B3 pini lojik-0 yapılır.

2.1.6. OUTPUT_TOGGLE( ) Komutu

Bu komut ile istenen portun istenen pini durum değiştirir (terslenir). Eğer pinin değeri lojik-1 ise lojik-0, lojik-0 ise lojik-1 olur. Kullanımı;

output_toggle(pin ismi); şeklindedir.

output_toggle(pin_a3); // A portunun A3 pininin değeri terslenir.
output_toggle(pin_b7); // B portunun B7 pininin değeri terslenir.

2.1.7. OUTPUT_BIT( ) Komutu

Bu komut ile istenen portun istenen pininin değeri değiştirilir. Pinin değeri parantez içerisine yazılır. Kullanımı;

output_bit(pin ismi, deger); şeklindedir.

output_bit(pin_a1,1); // A1 pinin değerini lojik-1 yapar.
output_bit(pin_b2,0); // B2 pinin değerini lojik-0 yapar.

2.1.8. INPUT_X( ) Komutu

Bu komut ile giriş olarak ayarlanan portun değeri okunur. “X” yerine değeri okunacak giriş portunun adı yazılır. Kullanımı;

deger=input_port ismi(); şeklindedir.

x=input_a(); // A portunun değeri okunarak x değişkenine aktarılır.
deger=input_b(); // B portunun değeri okunarak değer değişkenine aktarılır.

2.1.9. INPUT( ) Komutu

Bu komut ile istenen portun istenen pininin değeri okunur. Parantez içerisine pin adı yazılır. Kullanımı;

deger=input(pin ismi); şeklindedir.

deger=input(pin_a2); // A2 pininin değeri okunarak değer değişkenine aktarılır.
x=input(pin_c0); // C0 pininin değeri okunarak x değişkenine aktarılır.

2.1.10. INPUT_SATATE( ) Komutu

Bu komut ile istenen pinin giriş olarak mı yoksa çıkış olarak mı yönlendirildiği öğrenilir. “1” bilgisi ile geri dönerse giriş, “0” bilgisi ile geri dönerse pin çıkış olarak yönlendirilmiştir. Kullanımı;

deger=input_state(pin ismi); şeklindedir.

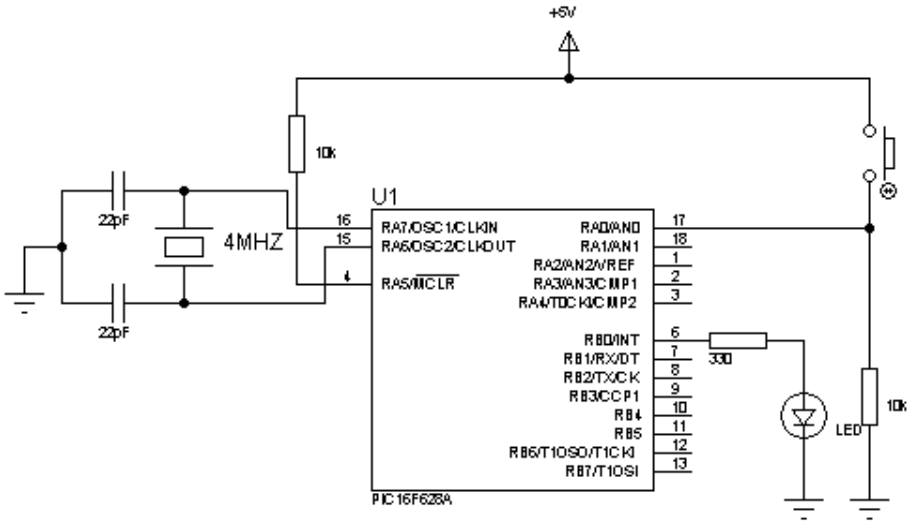
değer=input_state(pin_a3); // değer değişkenine A3 pini giriş olarak yönlendirildiyse “1”, çıkış olarak yönlendirildiyse “0” değerini atar.

2.1.11. Örnek Programlar

Örnek programlara ait proje dosyalarının oluşturulmasında PIC Wizard kullanılmıştır. Bu nedenle denetleyici konfigürasyonları görünmemektedir. Eğer manuel olarak proje dosyası oluşturursanız bu konfigürasyonları “#include C:\.....\.....” ifadesinden sonra yazmanız gerekir. Aşağıda örnek konfigürasyon ve gecikme fonksiyonları için gerekli olan osilatör frekans değerinin belirtilmesine örnek verilmiştir.

#fuses
XT,NOWDT,NOPROTECT,NOBROWNOUT,NOLVP,NOPUT,NOWRT,NODEBUG,NOCPD
#use delay(clock=4000000)

Örnek 1: Denetleyicinin A0 pinine bağlı olan butona basıldığında denetleyicinin B0 pinine bağlı olan LED’i yakan program;



Şekil 2.1: Bir butonla LED yakma devresi

DERLEYİCİ C Programı:

```
#include “D:\ISIS\giris_cikis.h”
#use fast_io(a)            // Programda A portunun kullanılacağı bildiriliyor.
```

```
#use fast_io(b)            // Programda B portunun kullanılacağı bildiriliyor.
void main( )     // Ana fonksiyon başlangıcı
{
     setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
     setup_timer_1(T1_DISABLED);
     setup_timer_2(T2_DISABLED,0,1);
     setup_comparator(NC_NC_NC_NC);
     setup_vref(FALSE);
     set_tris_a(0x01);   // 0x01 =0b00000001 olduğundan A portunun A0 pini giriş, diğer
     pinleri çıkış
                         // olarak yönlendiriliyor.
     set_tris_b(0x00);   // 0x00=0b00000000 olduğundan B portunun tüm pinleri çıkış olarak
                         // yönlendiriliyor.
     output_b(0x00);     // B portunun tüm pinleri lojik-0 (GND) seviyesine çekiliyor.
     while(1)    // Sonsun döngü
     {
        if(input(pin_a0)==1)     // A0 pin değeri okunarak lojik-1 olup olmadığına bakılıyor.
        Eğer
        {                        // lojik-1ise(buton basılı demek) output_high(pin_b0)
komutu çalışır.
           output_high(pin_b0);          // B0 pinini lojik-1 (+5 V) seviyesine çeker.
        }
     } // sonsuz döngü sonu
} // Ana fonksiyon sonu
```

Programda kullanılan her bir komutun görevi "//" işaretinden sonra açıklanmıştır. DERLEYİCİ C derleyicisi "//" operatörünü gördüğünde bundan sonra gelen ifadenin açıklama metni olduğunu anlar ve derleme işlemine dâhil etmez.

Programda kullanılan "# use fast_io();" komutları derleyiciyi A ve B portlarının program içinde giriş/çıkış olarak kullanılacağını bildirir. "set_tris_a(0x01); ve set_tris_b(0x00)" komutları ile denetleyici pinlerinin hangilerinin giriş, hangilerinin çıkış olacağı derleyiciyi bildirilir. Bu programda sadece A0 ve B0 pinleri kullanılacağından ve A0 pinine bağlı butonun durumuna gore B0 pinine bağlı LED'in yanması sağlanacağından dolayı A0 pininin giriş, B0 pininin çıkış olarak yönlendirilmesi gerekmektedir. Diğer pinlerin durumları şimdilik bizim için önemli değil. A0 pini giriş olacağından bu pine karşılık gelen bitin "1" olması, çıkış olarak kullanılacak B0 pinine karşılık gelen bitin "0" olması gerekmektedir. "output_b(0x00);" komutu ile devreye enerji verildiğinde LED'in yanmaması için çıkış olarak yönlendirilen B portunun tüm pinlerin lojik-0 seviyesine çekilmiştir. "if(input(pin_a0)==1);" komutu ile A0 pininin değeri okunmakta ve okunan değer "1"e eşit olduğunda "if" komutunun altındaki "{ }" parantezleri içine yazılan komut/komutların işlenmesi sağlanmaktadır. Butona basılmadığı sürece A0 pinin değeri "0"dır. Butona basıldığında direnç üzerinde gerilim düşümü olacağından A0 pininin değeri "1" olacak ve "output_high(pin_b0);" komutu çalışacaktır. Bu komut B0 pininin değerini lojik-1 (+5V) seviyesine çekecek ve LED ışık verecektir.

Örnek 2: Denetleyicinin A0 pinine bağlı olan butona basıldığında denetleyicinin B0 pinine bağlı olan LED yanıksa söndüren, sönükse yakan program;

Bu program için Resim 2'de verilen devre kullanılacaktır. Devreye ait program aşağıdaki gibidir.

DERLEYİCİ C Programı:

```
#include "D:\ISIS\pin_Tersle.h"
#use fast_io(a)          // Programda A portunun kullanılacağı bildiriliyor.
#use fast_io(b)          // Programda B portunun kullanılacağı bildiriliyor.

void main( )     // Ana fonksiyon başlangıcı
{
    setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
    setup_timer_1(T1_DISABLED);
    setup_timer_2(T2_DISABLED,0,1);
    setup_comparator(NC_NC_NC_NC);
    setup_vref(FALSE);

    set_tris_a(0x01);   // 0x01 =0b00000001 olduğundan A portunun A0 pini giriş, diğer
    pinleri çıkış
                        // olarak yönlendiriliyor.
    set_tris_b(0x00);   // 0x00=0b00000000 olduğundan B portunun tüm pinleri çıkış olarak
                        // yönlendiriliyor.
    output_b(0x00);     // B portunun tüm pinleri lojik-0 (GND) seviyesine çekiliyor.

    while(1)    // Sonsun döngü
    {
       if(input(pin_a0)==1)     // A0 pin değeri okunarak lojik-1 olup olmadığına bakılıyor.
       Eğer
       {                        // lojik-1ise(buton basılı demek) output_toggle(pin_b0)
komutu çalışır.
          delay_ms(200);        // buton arklarını önlemek için 200ms gecikme veriliyor.
          output_toggle(pin_b0);        // B0 pinini tersler (lojik-0 ise lojik-1, lojik-1 ise
          lojik-0 yapar.)
       }
    } // sonsuz döngü sonu
} // Ana fonksiyon sonu
```

Bu programda örnek 1'den farklı olarak "output_high(pin_b0);" komutu yerine pinin değerini terslemesi için "output_toggle(pin_b0);" komutu ve butona basıldığında oluşan arkları önlemek için "delay_ms(200);" komutu ile 200 mili saniyelik gecikme verilmiştir. Programın çalışması bu değişikliklerin dışında örnek 1 ile aynıdır.

2.2. Karar Verme Döngüleri ve Kullanım Örnekleri

Karar verme komutları; bir şarta bağlı olarak istenen işlemi gerçekleştiren komutlardır. Şartın doğruluğu ya da yanlışlığı şart deyiminin mantıksal yorumuyla ilgilidir. Bu konuya geçmeden önce 1.2.1.2 başlığı altında bulunan Örnek 3'e göz atmanız iyi olacaktır. DERLEYİCİ C dilinde kullanılan karar verme ifadeleri if, if-else, switch-case, while, do-while ve for komutlarıdır.

2.2.1. IF Komutu

Türkçe karşılığı "eğer" olan "if" komutu verilen şarta bağlı olarak kod ya da kod bloğunun çalışmasını sağlar. Şarta bağlı olarak bir tek kod çalıştırılacaksa bu kod "if" komutunun alt satırına yazılır. Şarta bağlı olarak birden fazla kod çalıştırılacaksa bu durumda "if" komutundan sonra "{" parantezinden sonra kodlar sırasıyla yazılarak "}" parantezi ile kod bloğu oluşturulur. Ancak "if" komutundan sonra "{ }" parantezleri kullanmayı alışkanlık haline getirmeniz karmaşık programlar yazdığınızda hata yapmanızı azaltacak ve programın anlaşılırlığını arttıracaktır.

```
if(şart)                    if(şart)
    komut;                  {
                                Komut_1;
                                Komut_2;
                                …
                            }
```

Dikkat ettiyseniz "if" komutundan sonra noktalı vigül işareti kullanılmamıştır.

```
if(sayi==3)          // sayi değişkeninin değeri "3" ise parantez içerisi TRUE
{                    // olur ve {  } parantezleri arasındaki kodları çalıştırır.
  output_high(pin_b3);   // sayı değişkeninin değeri "3" den farklı ise parantez içerisi
  delay_ms(500);         // FALSE olur ve parantez içerisindeki kodlar çalıştırılmaz.
"}"
  output_low(pin_b3);    // parantezinden sonraki kodları (varsa) çalıştırır.
}
```

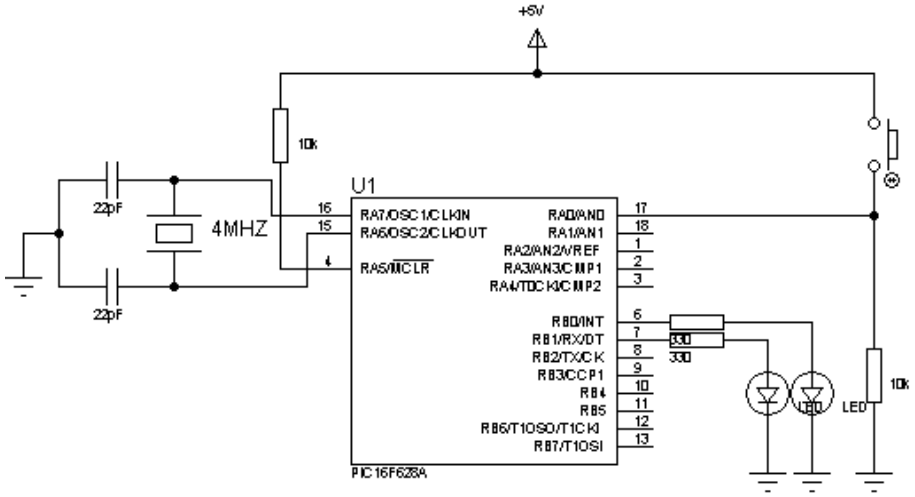
"if" komutunun kullanımı ve çalışması 2.1.11 konu başlığı altında verilen örnekte anlatılmıştır.

2.2.2. IF-ELSE Komutu

"if" komutuyla aynı şekilde çalışır. Birden fazla şartın olduğu uygulamalarda tercih edilir. Örneğin bir öğrenci için bir dersin yıl sonu notu 45 ve üzeri ise geçti, 45'in altında ise kaldı yazdıracak programı düşünelim. "if" komutu ile sadece "Geçti" ya da "Kaldı" yazdırabiliriz. İki durumuda yazdırabilmek için ayrı ayrı iki "if" komutu kullanmamız gerekir (birinde 45 ve üzeri ise "Geçti" yaz, diğer "if" komutunda ise 45'in altındaysa "Kaldı" yaz). Böyle durumlarda "if-else" komutu yardımımıza koşar. Komutun kullanımı aşağıda verilmiştir;

```
if(şart)            // şart doğru ise "if" den sonraki "{ }" parantezleri içindeki
{                   // komutları çalıştırır.
  Komut ya da komutlar;
}else               // şart yanlış ise "else" den sonraki "{ }" parantezleri içindeki
{                   // komutları çalıştırır.
  Komut ya da komutlar;
}
```

Örnek: Aşağıda verilen devrede butona basılı olduğu sure boyunca RB0 pinine bağlı LED'i, butona basılmadığı sure boyunca da RB1 pinine bağlı LED'i yakan programı yazalım.



Şekil 2.2: Bir butonla LED'lerin durumunu değiştirme

Görüldüğü gibi burada "buton basılıysa" ve "buton basılı değilse" olmak üzere iki şart var ve her şarta göre farklı bir işlemin yapılması gerekiyor.

DERLEYİCİ CProgramı:

```
#include "D:\ISIS\if_else.h"
#use fast_io(a)         // Programda A portunun kullanılacağı bildiriliyor.
#use fast_io(b)         // Programda B portunun kullanılacağı bildiriliyor.

void main( )    // Ana fonksiyon başlangıcı
{
    setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
    setup_timer_1(T1_DISABLED);
    setup_timer_2(T2_DISABLED,0,1);
    setup_comparator(NC_NC_NC_NC);
    setup_vref(FALSE);
```

```
    set_tris_a(0x01);   // 0x01 =0b00000001 olduğundan A portunun A0 pini giriş, diğer
    pinleri çıkış
                        // olarak yönlendiriliyor.
    set_tris_b(0x00);   // 0x00=0b00000000 olduğundan B portunun tüm pinleri çıkış olarak
                        // yönlendiriliyor.
    output_b(0x00);     // B portunun tüm pinleri lojik-0 (GND) seviyesine çekiliyor.

    while(1)    // Sonsun döngü
    {
       if(input(pin_a0)==1)     // A0 pin değeri okunarak lojik-1 olup olmadığına bakılıyor.
       Eğer
       {                        // lojik-1ise(buton basılı demek) output_togglepin_b0)
komutu çalışır.
          output_high(pin_b0);          // B0 pinini lojik-1 yapar.
          output_low(pin_b1);           // B1 pinini lojik-0 yapar.
       }else
       {
          output_low(pin_b0);           // B0 pinini lojik-0 yapar.
          output_high(pin_b1);          // B1 pinini lojik-1 yapar.
       }
    } // sonsuz döngü sonu
} // Ana fonksiyon sonu
```

Bu programda; “if-else” deyimindeki “if(input(pin_a0)==1)” ifadesiyle butona basılıysa B0 pinini lojik-1 ve B1 pinini lojik-0 yapılmakta, “else” ifadesiyle de butona basılı değilse B0 pini lojik-0, B1 pini lojik-1 yapılmaktadır. Böylece iki durum içinde farklı işlemleri gerçekleştirilebilmektedir.

2.2.3. SWITCH – CASE Komutu

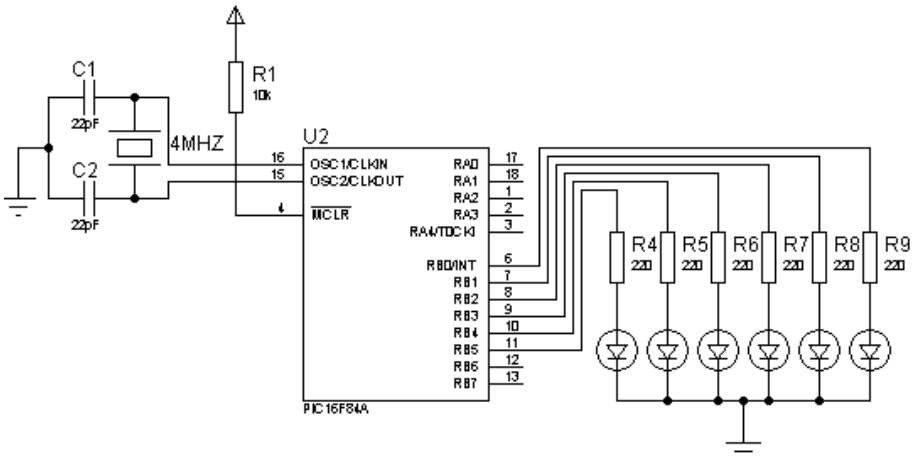
Bir değişkenin birden fazla durum alma olasılığının olduğu programlarda tercih edilen bir karar verme komutudur. Örneğin programımızda **not** adında bir değişkenimiz olsun. Bildiğimiz gibi bu değişken 0-5 aralığında 6 farklı değer alabilir. Böyle bir durumda **not** değişkeninin değeriyle ilgili bir program yazmak gerektiğinde “switch-case” komutu programcıya kolaylık sağlayacaktır. Komutun kullanımı;

```
switch ( ifade ya da değişken )
{
   case durum_1: komut veya komutlar;  break;
   case durum_2: komut veya komutlar; break;
   …
   default: komut veya komutlar; break;
}
```

İfade veya değişkenin değeri verilen durumlardan hangisine eşitse o durumdan sonraki komut/komutlar icra edilir ve “break;” deyimiyle “switch-case” komutu sonlandırılır. Eğer

ifade ya da değişken verilen durumlardan hiç birine uymuyorsa “default :” ifadesinden sonra yazılan komut/komutlar icra edilerek, “break;” deyimiyle komut sonlandırılır.

Örnek: Program yazım aşamasında girilen nota karşılık gelen pine bağlı LED’i yakan program yazalım (not=0 sa B0’ a, not=1 ise B1’ e, ….not=5 ise B5’ e bağlı LED).



Şekil 2.3: Not değerine karşılık gelen LED’i yakan devre

DERLEYİCİ C Programı:

```
#include “D:\ISIS\not_goster.h”
#use fast_io(b)          // Programda B portunun kullanılacağı bildiriliyor.
int not=3;               // Tamsayı türünde not adında değişken tanımlanıyor ve 3 değeri
atanıyor.
void main( )             // Ana fonksiyon başlangıcı
{
    setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
    setup_timer_1(T1_DISABLED);
    setup_timer_2(T2_DISABLED,0,1);
    setup_comparator(NC_NC_NC_NC);
    setup_vref(FALSE);

    set_tris_b(0x00);   // 0x00=0b00000000 olduğundan B portunun tüm pinleri çıkış olarak
                        // yönlendiriliyor.
    output_b(0x00);     // B portunun tüm pinleri lojik-0 (GND) seviyesine çekiliyor.
    switch(not)
    {
      case 0: output_b(0x01); break;  // not=0 ise B0 pinine bağlı LED’i yakar.
      case 1: output_b(0x02); break;  // not=1 ise B1 pinine bağlı LED’i yakar.
      case 2: output_b(0x04); break;  // not=2 ise B2 pinine bağlı LED’i yakar.
      case 3: output_b(0x08); break;  // not=3 ise B3 pinine bağlı LED’i yakar.
      case 4: output_b(0x10); break;  // not=4 ise B4 pinine bağlı LED’i yakar.
      case 5: output_b(0x20); break;  // not=5 ise B5 pinine bağlı LED’i yakar.
```

```
        default: output_b(0x3F); break; // not değişkeni 0…5 aralığından farklı bir değer ise
        tüm
                                                    // LED'leri yakar.
    }
}
```

"int not=3;" ifadesi ile tamsayı türünde **not** isminde bir değişken tanımlanmış ve değer olarak **3** atanmıştır.

"Switc-Case" ifadesinde ise **not** değişkeni kullanılmış ve bu değişkenin alabileceği durumlar ve bu durumlarda yapılacak işlemlere ait komutlar kullanılmıştır. **Not** değişkeninin değeri 3 olduğundan program "case 3:" ifadesinden sonraki komutları icra eder. Burada "output_b(0x08); komutu ile B portuna binary olarak 0b00001000 bilgisi gönderilmekte yani B3 pinine bağlı LED yakılmakta ve "break;" komutu ile "switch-case" yapısı sonlandırılmaktadır. Eğer **not** değişkeninin değerei 0-5 aralığından farklı bir değer alsaydı o zaman "default :" ifadesinden sonraki komut icra edilerek, "output_b(0x3F);" komutu ile (0b00111111) B portuna bağlı tüm ledlerin (B0 – B5) yanması sağlanacak ve hata olduğu anlaşılacaktır.

2.2.4. WHILE Komutu

Bir şarta bağlı olarak programdaki bir kodun ya da kod bloğunun tekrar etmesi istendiğinde kullanılan komuttur. Kullanımı;

```
while ( şart )
{
  Komut ya da komutlar;
}
```

Örnek: A portuna bağlı olan butona basıldığı sürece B portuna bağlı olanLED'i yakan, buton bırakıldığında LED'i söndüren programı yazalım.

Elektronik devre olarak Resim-2.1 bir butonla LED yakma devresi kullanılacaktır.

DERLEYİCİ C Programı:

```
#include "D:\ISIS\while_kullanimi.h"
#use fast_io(a)          // Programda A portunun kullanılacağı bildiriliyor.
#use fast_io(b)          // Programda B portunun kullanılacağı bildiriliyor.

void main( )     // Ana fonksiyon başlangıcı
{
    setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
    setup_timer_1(T1_DISABLED);
    setup_timer_2(T2_DISABLED,0,1);
    setup_comparator(NC_NC_NC_NC);
    setup_vref(FALSE);

    set_tris_a(0x01);   // 0x01 =0b00000001 olduğundan A portunun A0 pini giriş, diğer
    pinleri çıkış
```

```
                        // olarak yönlendiriliyor.
    set_tris_b(0x00);   // 0x00=0b00000000 olduğundan B portunun tüm pinleri çıkış olarak
                        // yönlendiriliyor.
    output_b(0x00);     // B portunun tüm pinleri lojik-0 (GND) seviyesine çekiliyor.

    while(1)    // Sonsun döngü
    {
        while(input(pin_a0)==1)         //  A0 pinindeki buton basılı olduğu sürece dön.
        {
            output_high(pin_b0);        // B0 pinini lojik-1 yaparak bağlı LED'i yakar.
        }
      output_low(pin_b0);               // B0 pinini lojik-0 yapar ve LED söner.
    }
}
```

Programda görüldüğü üzere iki farklı "while" döngüsü kullanılmıştır. Bunlardan ilki "while(1)" sonsuz döngüdür. Çünkü kullanıcının butona ne zaman basacağı/bırakacağı belli olmadığı için enerji olduğu sürece çalışmanın tekrarlanması gerekmektedir. Burada parantez içerisine yazılan "1" rakamı mantıksal olarak TRUE anlamı taşımakta ve şart doğru olduğu için (Şart doğru olduğunda TRUE değeri döndürür.) döngünün sürekli çalışmasını sağlamaktadır.

İkinci "while" komutunda ise şart olarak A0 pini kontrol edilmekte ve "1'e eşit mi?" diye bakılmaktadır. "while(input(pin_a0)==1)" butona basıldığında A0 pini lojik-1 değerinde olacağından şart gerçekleşecek ve bu döngüye ait kod bloğu içinde bulunan kod/kodlar şart sağlandığı sürece tekrarlanacaktır. Bu durumda "output_high(pin_b0);" komutu ile B0 pini lojik-1 yapılarak bağlı olan LED'in yanması sağlanmıştır.

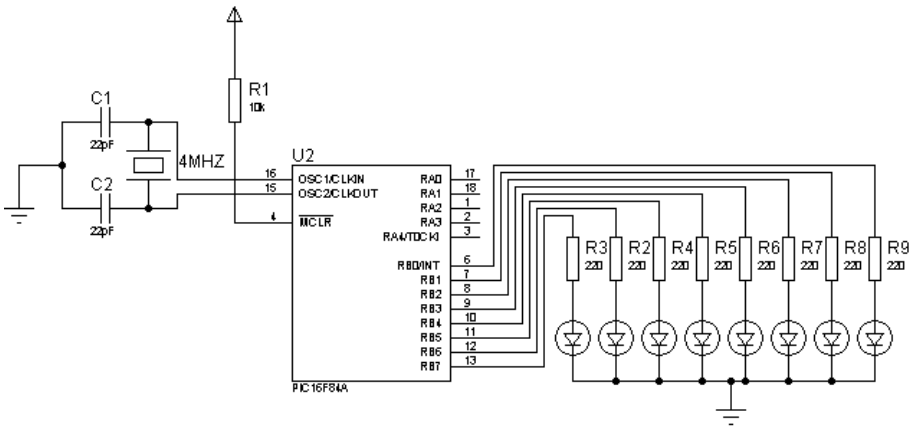
Şart gerçekleşmediğinde (buton bırakıldığında) döngü sonlanarak döngüye ait kod bloğunun altındaki kod/kodlar icra edilecektir. Bu durumda "output_low(pin_b0);" komutu ile B0 pini lojik-0 yapılarak LED'in sönmesi sağlanır.

2.2.5. DO-WHILE Komutu

While komutundan farklı olarak şart sonda kontrol edilir. Bu yüzden döngü en az bir kere çalışır. Bu özelliğinin dışında "while" komutu ile aynı özelliklerdedir. Kullanımı;

```
do
{
  Komut ya da komutlar;
} while (şart);
```

Örnek: Devreye enerji verildiğinde B portuna bağlı LED'leri sırayla yakan ve B7 pinindeki LED yandığında duran programı yazalım.



Şekil 2.4: Yürüyen ışık devresi

DERLEYİCİ C Programı:

```
#include "D:\ISIS\do_while_yuruyen_isik.h"
#use fast_io(b)          // Programda B portunun kullanılacağı bildiriliyor.
long int a=1;            // a adında 16 bitlik tamsayı tanımlanıyor ve 1 değeri atanıyor.
void main( )     // Ana fonksiyon başlangıcı
{
    setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);

    set_tris_b(0x00);   // 0x00=0b00000000 olduğundan B portunun tüm pinleri çıkış olarak
                        // yönlendiriliyor.
    output_b(0x00);     // B portunun tüm pinleri lojik-0 (GND) seviyesine çekiliyor.
    do
    {
       output_b(a);     // a değişkeninin değeri B portuna gönderiliyor.
       a=a<<1;          // a içeriği bit bazında 1 bit sola öteleniyor.
       delay_ms(200);   // 200ms gecikme veriliyor.
    }while(a<=0x80);

}
```

Program derleyerek denetleyiciye yükleyin ve devreye enerji verin. Soldan başlayarak sırayla sağa doğru LED'lerin yandığını ve en sağdaki (RB7 ye bağlı) LED yandığında programın durduğunu göreceksiniz. Bunun sebebi "do-while" döngüsünde kullanılan şart ifadesidir (a<=0x80). Bu ifade ile a değişkeninin değeri binary olarak 0b10000000 oluncaya kadar (RB7 ye bağlı LED yanıncaya kadar) döngü devam ettirilmiştir.

Program çalışmaya başladığında a değişkeninin binary olarak değeri 0b00000001'dir. Herhangi bir şarta bakılmaksızın döngü çalışmış ve bu değer "output_b(a);" komutu ile bu değer çıkışa (B portuna) gönderilir. "a=a<<1;" ifadesi ile sola öteleme yapılır ve a

değişkeninin değeri 0b00000010 olur, 100 ms'lik duraklama verilerek şart kontrol edilir. *a*değişkeninin değeri (0b00000010) 0x80 sayısından (0b10000000) küçük ya da bu sayıya eşit ise döngü devam eder ve a değişkeninin yeni değeri çıkışa gönderilir. Bu işlemler a değişkeninin değeri 0x80 (0b10000000) olduğunda son kez tekrar eder ve RB7 ye bağlı LED yanar. Bu tekrar olayında a değişkeninin değeri sola ötelemeden dolayı 8 bitlik sınırların dışına çıkar ve (0b0000000 100000000) olur. Bu sebepten dolayı a değişkeni long int (16 bit'lik tamsayı) türünde tanımlanmıştır.

2.2.6. FOR Komutu

"for" döngüsü döngü işleminin belirli sayıda yapılması gereken yerlerde kullanılır. Bu döngü komutunun diğerlerinden farkı bir sayaç alması ve bu sayaç yardımı ile döngü sayısının kontrol edilmesidir. Belirli bir başlangıç değerinden başlayarak, belirli bir değere ulaşıncaya kadar istenen adımlarda sayaç değişkeninin değerini arttırır/azaltır. Böylece döngü sayısı kontrol edilir.

```
for( başlangıç değeri; bitiş değeri; adım sayısı)
{
  Komut ya da komutlar;
}
```

For döngüsünün kullanımıyla ilgili çalışma 1.2.1.2. "Operatörler ve Örnek Programlar" konusunda bulunan Örnek 6'da yapılmıştır. Bu konuyu geçmeden önce lütfen örnek 6'yı tekrar okuyunuz.

2.3. Gecikme Fonksiyonları ve Örnek Program Yazılımları

Program icra edilirken programı belirli bir sure durduran komutlara gecikme fonksiyonu denir. Gecikme fonksiyonu derleyiciye programın başında "#use delay_(clock=osilatör frekansı)" şeklinde bir ifadeyle bildirilmelidir. Bu ifadede bulunan osilatör frekansı yerine devrede kullanılacak osilatör frekansı (4000000 ya da20000000) Hz cinsinden yazılır.

#use delay(clock=4000000)

2.3.1. DELAY_US( ) Komutu

Mikro saniye cinsinde gecikme sağlayan fonksiyondur. Parantez içerisine tamsayı ya da değişken yazılabilir. Kullanımı;

delay_us(gecikme miktarı); şeklindedir.

delay_us(200); // 200 mikro saniye gecikme sağlar.

2.3.2. DELAY_MS( ) Komutu

Milisaniye cinsinden gecikme sağlayan fonksiyondur. Parantez içerisine tamsayı ya da değişken yazılabilir. Kullanımı;

delay_ms(gecikme miktarı); şeklindedir. Şekil 2.10 da verilen Do-While döngüsü ile yürüyen ışık programında bu fonksiyon kullanılmıştır.

delay_ms(200); // 200 mili saniye gecikme sağlar.

2.3.3. DELAY_CYCLES( ) Komutu

İstenen komut sayısı kadar gecikme verir. Parantez içerisini 1-255 arası bir tamsayı değer girilir. Kullanımı; **delay_cycles(komut sayısı);** şeklindedir.

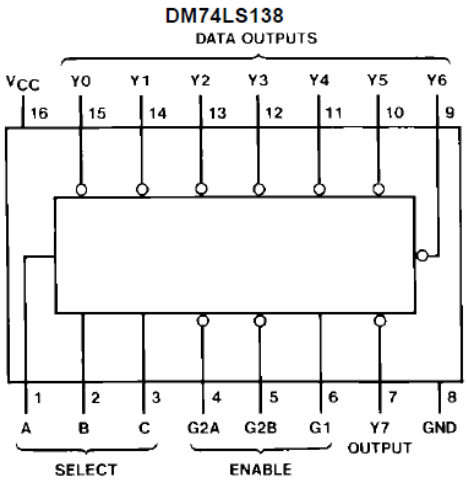
delay_cycles(20); // 20 komut işlenme süresi kadar gecikme sağlar.

2.4. Port Çoğullama ve Matris Display Uygulamaları

Bazı devrelerde denetleyici pin sayısı yetmeyebilir. Böyle durumlarda port çoğullama yapmak gerekir. Port çoğullama işlemini yapmak için özel entegreler kullanılır. Bu entegrelerden piyasada rahatlıkla bulunabilenlerden bazıları 74138, 74164,8255 ve 74HC595 entegreleridir. Bu bölümde sadece 74138 entegresi üzerinde duracağız. Port çoğullamanın mantığı aynıdır sadece kullanılacak entegrenin özellikleri değişmektedir. Diğer entegrelerin kullanımını üretici bilgi sayfalarına bakarak kolayca uygulama yapabilirsiniz.

2.4.1. 74138 Entegresi ile Port Çoğullama

74138 entegresi 3 bitlik kontrol bilgisi ile 8 bitlik çıkış verebilen bir entegredir. Dolayısıyla denetleyicinin sadece 3 pinini kullanarak 8 pin elde etmiş olursunuz. 74138 entegresinin 3 adet kontrol ucu vardır. Bu uçlar sayesinde istenen sayıda 74138 entegresi ard arda bağlanarak çok sayıda çıkış elde edebilirsiniz. Aşağıda bu entegrenin pin diyagramı (Şekil 2.5) ve fonksiyon tablosu (Şekil 2.6) verilmiştir.



Şekil 2.5: 74LS138 Pin Diyagramı

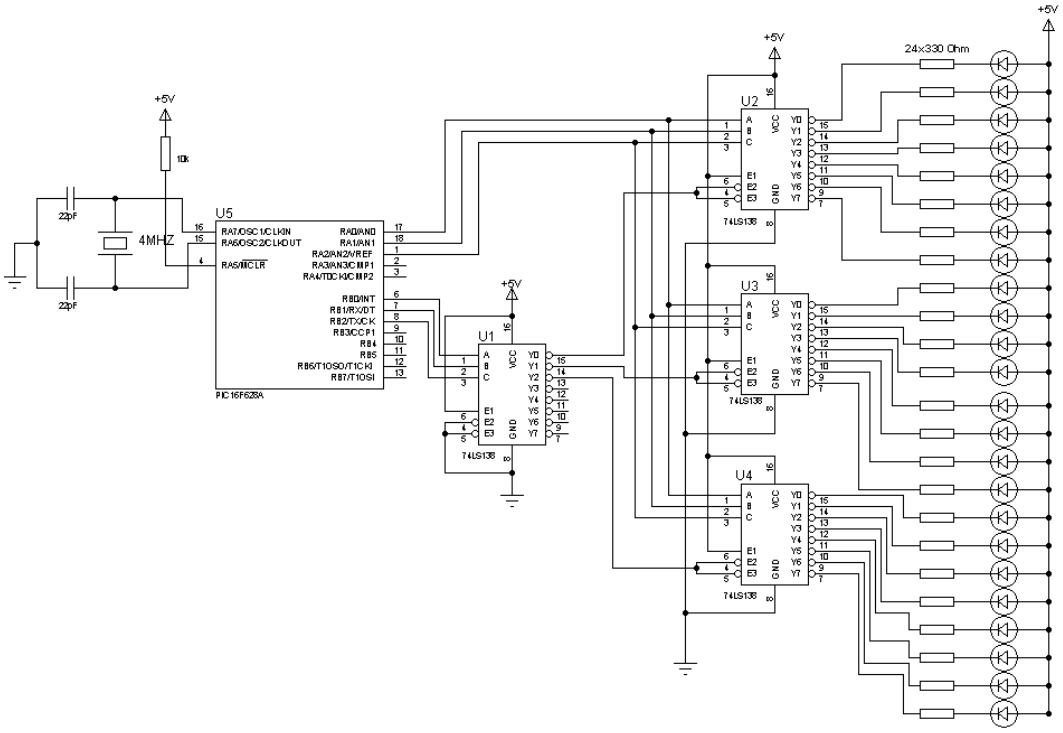
DM74LS138

Inputs					Outputs							
Enable		Select										
G1	G2 (Note 1)	C	B	A	YO	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

Şekil 2.6: 74LS138 Fonksiyon Tablosu

Pin diyagramında görülen A, B ve C uçları kontrol uçları, G2A, G2B ve G1 uçları ise entegre seçme uçlarıdır. Diyagrama dikkatle baktığımızda G2A ve G2B pinlerinin lojik-0 da, G1 pininin ise lojik-1 de aktif olduğu görülür. Y0-Y7 uçları ise entegrenin çıkış uçlarıdır. Fonksiyon tablosuna bakarak A, B ve C uçlarından hangi kontrol bilgisi gönderildiğinde hangi çıkışların durumunun ne olacağı görülmektedir. Örneğin C=0, B=1 ve A=1 bilgisi (011 ya da L-H-H) gönderildiğinde Y3 çıkışının lojik-0 (L), diğer çıkışların lojik-1 (H) olduğu görülmektedir.

Örnek: Aşağıda verilen 74138 entegresi ile port çoğullama devresinde LED'lerin sırayla yanmasını sağlayan programı yazalım.



Şekil 2.7: 74138 entegresi ile yürüyen ışık devresi

DERLEYİCİ CProgramı:

```
#include "D:\ISIS\_74138_yuruyenIsik.h"
#use fast_io(a)          // Programda A portunun kullanılacağı bildiriliyor.
#use fast_io(b)          // Programda B portunun kullanılacağı bildiriliyor.
int cikis;               // cikis adında 8 bitlik tamsayı türünde değişken tanımlanıyor.
void main( )             // Ana fonksiyon başlangıcı
{
    setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
    setup_timer_1(T1_DISABLED);
    setup_timer_2(T2_DISABLED,0,1);
    setup_comparator(NC_NC_NC_NC);
    setup_vref(FALSE);

    set_tris_a(0x00);   // 0x01 =0b00000001 olduğundan A portunun A0 pini giriş, diğer
    pinleri çıkış
                        // olarak yönlendiriliyor.
    set_tris_b(0x00);   // 0x00=0b00000000 olduğundan B portunun tüm pinleri çıkış olarak
                        // yönlendiriliyor.
    while(1)
    {
      output_b(0x00);          // U2 entegresinin seçilmesini sağlar.
```

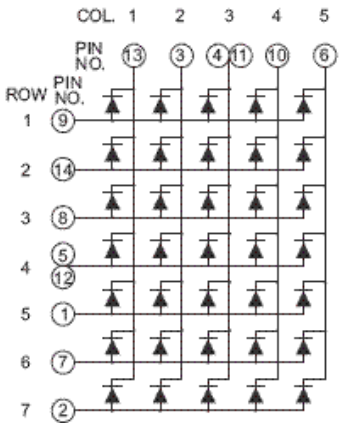
```
        for(cikis=0;cikis<=7;çıkış++)
        {
            output_a(çıkış);        // U2 çıkışlarına bağlı LED'leri çıkış değerine göre sırayla
            yakar.
            delay_ms(250);
        }
        output_b(0x01);             // U3 entegresinin seçilmesini sağlar.
        for(cikis=0;cikis<=7;çıkış++)
        {
            output_a(çıkış);        // U3 çıkışlarına bağlı LED'leri çıkış değerine göre sırayla
            yakar.
            delay_ms(250);
        }
        output_b(0x02);             // U4 entegresinin seçilmesini sağlar.
        for(cikis=0;çıkış<=7;çıkış++)
        {
            output_a(çıkış);        // U4 çıkışlarına bağlı LED'leri çıkış değerine göre sırayla
            yakar.
            delay_ms(250);
        }

    }
}
```

Devreye bakıldığında denetleyicinin A portu U2, U3 ve U4 entegrelerinin kontrol uçlarına (A,B,C) gitmiştir. Yani A portu ile gönderilen bilginin durumuna göre bu entegrelerin çıkışları değişir. Denetleyicinin B portu ise U1 entegresinin kontrol uçlarına gitmiştir. Yani U1 entegresinin çıkışları B portundan gönderilen bilgi ile değişmektedir. U1 entegresinin çıkışları ise U2, U3 ve U4 entegrelerinin entegre seçme ucu olan G2A ve G2B uçlarına gitmektedir. Dolayısıyla U2, U3 ve U4 entegrelerinden hangisinin aktif olacağı B portundan gönderilen bilgiye göre değişmektedir.

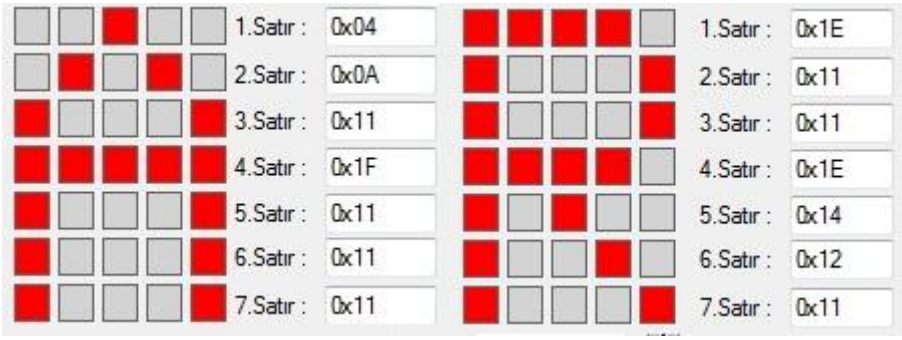
U2 entegresinin seçilmesi için U1 entegresinin Y0 çıkışının lojik-0 (L) olması gerekir. Fonksiyon tablosuna baktığımızda bunun için U1 entegresinin (C,B ve A) uçlarına B portundan sıfır bilgisi gönderilmelidir. "output_b(0x00);" komutu ile bu işlem gerçekleştirilmiş ve U1' in Y0 çıkışı lojik-0 yapılarak U2 entegresi seçilmiştir. U2 entegresinin 8 adet çıkışı olduğundan ve her çıkışında LED bulunduğundan "for(çıkış=0;çıkış<=7;çıkış++)" döngüsü ile 8 turluk bir döngü oluşturulmuş ve çıkış sayaç değeri her döngüde "output_a(çıkış);" ifadesiyle A portuna yani U2, U3 ve U4 entegrelerinin kontrol uçlarına gönderilmiştir. Bu durumda sadece U2 entegresi seçili olduğundan buna bağlı LED'ler sıra ile yanar. Daha sonra "output_b(0x01);" komutu ile U3 entegresi ve daha sonra da "output_b(0x02);" komutu ile U4 entegresi seçilerek aynı işlemler tekrar eder.

2.4.2. Matris Display Uygulamaları

Matris yapıda LED'lerden oluşan ve değişik karakter ya da görüntüler oluşturan görüntüleme aygıtıdır. Piyasada en çok 5x7 (5 sütun, 7 satır) ve 8x8 (8 sütun, 8 satır) displayler kullanılır. Matris displayler LED bağlantı şekline göre anot sütun ve katot sütun olmak üzere iki yapıda bulunurlar. Katot sütun bir display de sütuna lojik-0, satıra lojik-1, anot sütun bir displayde ise sütuna lojik-1, satıra lojik-0 uygulamak gerekir. Aşağıda matris display içyapısı ve görünüşü verilmiştir.



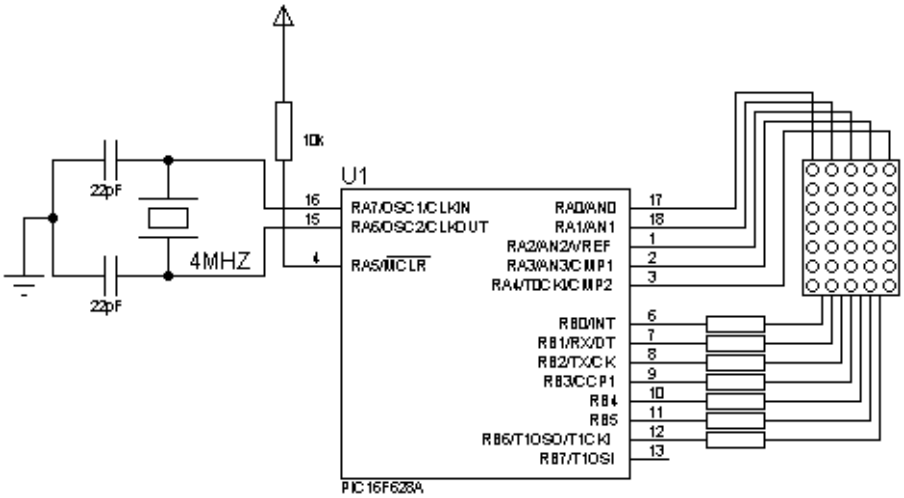
Şekil 2.8: Matris display iç yapısı **Şekil 2.9: Matris display görünüşü**



Şekil 2.10: Matris display'de karakterlerin oluşması ve karakter satır bilgileri

Matris display uygulamalarında ciddi manada port sıkıntısı olduğundan dolayı tarama yöntemi kullanılır. Tarama yönteminde 1.sütun aktif edilir ve o sütunda yanması istenen satır LED'lere ait satır bilgisi gönderilir. Daha sonra 2.sütun aktif edilerek o sütunda yanması istenen satır bilgisi gönderilir. Bu olay böyle devam eder. Saniyede 15 ve üzeri sayıda tekrar edildiğinde karakteri oluşturan tüm LED'ler aynı anda yanıyormuş gibi görünür.

Örnek: Tarama yöntemi ile matris displayda "DERLEYİCİ -C" yazan program



Şekil 2.11: Matris display uygulaması

DERLEYİCİ C Programı:

```
#include "D:\\ISIS\5x7_Matrix_Display.h"
#use fast_io(b)
#use fast_io(c)
#use fast_io(d)

int C[]={0xDD,0xBE,0xBE,0xBE,0xC1};          // C karakteri kodları
int S[]={0xCD,0xB6,0xB6,0xB6,0xD9};          // S karakteri kodları
int tire[]={0xFF,0xF7,0xF7,0xF7,0xFF};       // - karakteri kodları
int a,b;

int sutunSecimi(char sutun)       // Sütun seçiminin yapıldığı fonksiyon
{
  switch(sutun)
{
     case 0:return(0x10); break;
     case 1:return(0x08); break;
     case 2:return(0x04); break;
     case 3:return(0x02); break;
     case 4:return(0x01); break;
}
}

void karakterGoster(char karakter[]) // Displayde gösterme fonksiyonu
{
  for(a=0;a<20;a++)  //  Her karakteri 20 kere görüntüler
{
     for(b=0;b<5;b++)  // Karaktere ait satır bilgilerini (5 adet) sıra ile işler
{
        output_d(sutunSecimi(b));
        output_b(karakter[b]);
```

```
        delay_ms(10);
}
}
}
void main()      // Ana fonksiyon başlangıcı
{
  setup_adc_ports(NO_ANALOGS);
  setup_adc(ADC_OFF);
  setup_psp(PSP_DISABLED);
  setup_spi(SPI_SS_DISABLED);
  setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
  setup_timer_1(T1_DISABLED);
  setup_timer_2(T2_DISABLED,0,1);

set_tris_b(0x00);        // B portu komple çıkış
set_tris_c(0x00);        // C portu komple çıkış
set_tris_d(0x00);        // D portu komple çıkış

  while(1)       // Sonsuz döngü
{
    karakterGoster(C);          // C karakterini gönderir.
    karakterGoster(C);          // C karakterini gönderir.
    karakterGoster(S);          // S karakterini gönderir.
    karakterGoster(tire);       // - karakterini gönderir.
    karakterGoster(C);          // C karakterini gönderir.
}
}
```

2.5. Kesme (interrupt) İşlemleri

Programın icrası sırasında bazı giriş-çıkış faaliyetlerinin kontrol edilmesi gerekebilir. Böyle durumlar oluştuğunda kontrol işlemi program içerisinde yazılımsal olarak yapıldığında programın hızında ve performansında düşüş meydana gelir. Bu tarz giriş-çıkış faaliyetlerinin denetleyicinin donanımsal özelliği kullanılarak kontrol edilmesine **kesme** denir. Kesme işlemi gerçekleştiğinde program kaldığı yeri program counter **stack** kaydedicisine kaydeder ve program kesme fonksiyonuna gider. Kesme fonksiyonundaki işler tamamlandığında program counter, stack kaydedicisine kaydedilen değeri alarak programın kaldığı yerden devam etmesini sağlar. Her denetleyicinin sahip olduğu kesmeler farklılık göstermekle birlikte bunların çalışma mantığı hemen hemen aynıdır.

Örneğin 2.1.11. Örnek programlar konusunda anlatılan örnek 1'e baktığımızda butona basılıp basılmadığı sonsuz döngü içerisinde "if" komutu ile kontrol edilmiştir. Eğer bu program biraz daha uzun ve karmaşık olsaydı buton kısa süreli basıldığında denetleyici bunu algılamayabilirdi. Çünkü yazılımsal kontrol işleminde butonun basılı olduğu anda kontrol komutunun icra edilmesi şarttır.

DERLEYİCİ C de kesme fonksiyonu ana programdan önce tanımlanmalıdır. Fonksiyon tanımı;

#İnt_kesme adı
<Fonksiyon geri dönüş değer türü><Fonksiyon adı><()>
{
Komut ya da komutlar;
}

Örnek kullanımı;

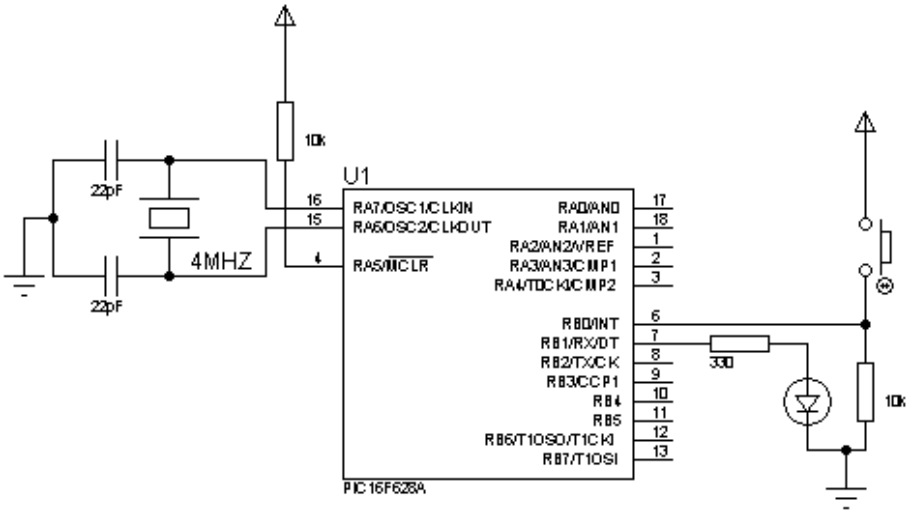
```
#int_ext
void harici_kesme()
{
  output_high(pin_b5);
  delay_ms(250);
  output_low(pin_b5);
}
```

Kesme fonksiyonu yazıldıktan sonra kesmelerin kullanılabilmesi için ana fonksiyon içinde kesmelere izin verilmelidir. Aksi halde kesme pasiftir ve kullanılamaz. Kesmeleri aktif/pasif yapmak için aşağıda verilen ifadeler kullanılır.

enable_interrupts(kesme adı); // Kesmeyi aktif yapar.
disable_interrupts(kesme adı); // Kesmeyi pasif yapar.

Aktif yapılan kesmelere izin vermek/izin vermemek için;
enable_interrupts(GLOBAL); // Aktif yapılan kesmelere izin verir.
disable_interrupts(GLOBAL); // Aktif yapılan kesmeye izin vermez.

Örnek: Dış kesme uygulaması



Şekil 2.12: Harici kesme uygulama devresi

```
#include "D:\ISIS\kesmesiz.h"

#use fast_io(b)
void main()
{

   setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DISABLED,0,1);
   setup_comparator(NC_NC_NC_NC);
   setup_vref(FALSE);

   set_tris_b(0x01);
   output_b(0x00);
   while(1)
   {
      if(input(pin_b0)==1)
      {
         output_high(pin_b1);
      }else
      {
         output_low(pin_b1);
      }
      delay_ms(1000);
   }

}
```

```
#include "D:\ISIS\harici_kesme.h"

#use fast_io(b)
#int_ext
void harici_kesme()
{
   output_high(pin_b1);
}
void main()
{

   setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DISABLED,0,1);
   setup_comparator(NC_NC_NC_NC);
   setup_vref(FALSE);

   set_tris_b(0x01);
   output_b(0x00);

   enable_interrupts(int_ext);
   enable_interrupts(GLOBAL);

   while(1)
   {
      output_low(pin_b1);
      delay_ms(1000);
   }

}
```

Yukarıda verilen programda sol bölümde butonun basılması yazılımsal olarak (kesme kullanılmadan) kontrol edilmektedir. Devreyi kurarak programı denediğinizde LED'in yanması için butona uzun süre basmanız gerektiğini göreceksiniz. Bunun nedeni kontrol işlemi yazılımsal olarak yapılmakta ve "delay_ms(1000);" komutu nedeniyle 1000 milisaniyede bir kontrol edilmektedir. Ancak sağdaki bölüme bakarsanız kesme fonksiyonu tanımlanmış ve LED bu fonksiyon içinde yakılmıştır. Kesme kullanmanın önemini vurgulamak için yine sonsuz döngü oluşturulmuş ve bu döngü içinde de 1000 ms'lik gecikme verilmiştir. Devreyi bu programı yükleyerek denediğinizde butona basar basmaz LED'in yandığını göreceksiniz.

2.6. Zamanlayıcı ve Sayıcı İşlemleri

Zamanlayıcı ve sayıcı işlemleri denetleyicinin aynı birimi tarafından yapılır. Aralarındaki fark zamanlayıcının sabit bir frekansı vardır ve bunu denetleyicinin kristalinden temin eder. Sayıcıda ise sabit bir frekans yoktur ve artış için gerekli olan sinyal dış kaynaktan alınır. Kesmelerde anlatıldığı gibi zamanlayıcı ve sayıcı işlemleri yazılımsal olarak yapıldığında denetleyiciyi meşgul eder. Bu yüzden denetleyicinin donanımsal özellikleri ile bu işlerin yapılması daha uygun olacaktır. Kullanılan denetleyicinin türüne bağlı olarak sahip olduğu zamanlayıcı/sayıcı birimi farklılık gösterebilir.

2.6.1. TIMER0 Birimi

Sinyal kaynağının seçimine bağlı olarak zamanlayıcı ve sayıcı olarak kullanılabilen 8 bitlik bir birimdir. Değeri 0xFF den 0x00 a geçtiğinde taşma kesmesi oluşturur. Başlangıçta Timer0 kaydedicisinin değeri 0'dır ve programlama aşamasında başlangıç değeri ayarlanabilir. Eğer bir başlangıç değeri atanırsa kesme fonksiyonu içinde de tekrar başlangıç değeri ayarlanmalıdır.

2.6.1.1. SETUP_TIMER_0() Komutu

Timer0 zamanlayıcısının ayarlarını yapar. Kullanımı; **setup_timer_0(mod);**şeklindedir. Mod kısmına aşağıda verilen özellikler yazılır ve her özellik arasına "|" işareti konur.

RTCC_INTERNAL : Timer0 frekans kaynağının denetleyici frekansı olduğunu belirtir.

RTCC_EXT_L_TO_H : Timer0 frekansının dış kaynaktan alınacağını ve yükselen kenarda tetikleneceğini belirtir.

RTCC_EXT_H_TO_L : Timer0 frekansının dış kaynaktan alınacağını ve düşen kenarda tetikleneceğini belirtir.

RTCC_DIV2, RTCC_DIV_4, RTCC_DIV_8, RTCC_DIV_16, RTCC_DIV_32, RTCC_DIV_64, RTCC_DIV_128, RTCC_DIV_256: frekansın sabitin sonunda belirtilen sayıya bölüneceğini belirtir.

Setup_timer_0(RTCC_DIV_32|RTCC_EXT_H_TO_L); İfadesi Timer0'ın sinyal kaynağı olarak harici kaynak kullanacağını (harici kaynak olduğundan sayıcı olarak kullanılacak), harici kaynak sinyalinin düşen kenarında artış olacağını ve bölme oranının 32 olacağını gösterir.

2.6.1.2. SET_TIMER_0( ) Komutu

Timer0 biriminin saymaya başlayacağı değeri ayarlar. Parantez içerisine 0-255 aralığında bir değer yazılabilir. Kullanımı;

set_timer0(deger); şeklindedir.

set_timer0(100); // 100 den saymaya başlar.

2.6.1.3. Timer0 Kesmesi

Taşma meydana geldiğinde oluşan kesmedir. Kesmeler konusunda anlatıldığı gibi kullanılır. "#int_kesme adı" yerine "#int_timer0" yazılır. Kesmenin oluşma süresi;

Süre = $\frac{fosc}{4}$ x Bölme oranı x (256-Başlangıç değeri) formolü ile hesaplanır.

2.6.2. TIMER1 Birimi

2.6.2.1. SETUP_TIMER1( ) Komutu

Timer1 zamanlayıcı ayarlarını yapar. Kullanımı; **setup_timer_1(mod);**şeklindedir. Mod kısmına aşağıda verilen tanımlamalar yazılır. Birden fazla tanımlama kullanılacaksa aralarına "|" işareti konur.

T1_DISABLED : Timer1'i kapat
T1_INTERNAL : Sinyal kaynağı denetleyici frekansı
T1_EXTERNAL : Sinyal kaynağı harici
T1_EXTERNAL_SYNC : Harici sinyal kaynağı ile senkronize
T1_CLK_OUT : Timer1 osilatörü kullanılacak

T1_DIV_BY_1, T1_DIV_BY_2, T1_DIV_BY_4, T1_DIV_BY_8: ifadenin sonundaki rakam bölme oranı olarak kullanılacak.

setup_timer_1(T1_EXTERNAL|T1_DIV_BY_2); // Timer1; harici sinyal kaynağını kullanacak ve bölme oranı 2 olacak.

2.6.2.2. SET_TIMER1( ) Komutu

Timer1 biriminin saymaya başlayacağı değeri ayarlar. Değer kısmına 0-65535 arası bir değer yazılabilir. Kesme kullanılacaksa kesme içinde bu komutla tekrar başlangıç değeri atanmalıdır. Kullanımı; **set_timer1(değer);** şeklindedir.

2.6.2.3. Timer1 Kesmesi

Taşma meydana geldiğinde oluşan kesmedir. Kesmeler konusunda anlatıldığı gibi kullanılır. "#int_kesme adı" yerine "#int_timer1" yazılır. Kesmenin oluşma süresi;

Süre = $\frac{fosc}{4}$ x bölme oranı x (65536-başlangıç değeri) formülü ile hesaplanır.

2.6.3. TIMER2 Birimi

2.6.3.1. SETUP_TIMER2( ) Komutu

Timer2 zamanlayıcı ayarlarını yapar. Kullanımı; **setup_timer_2(mod,peryot,postscaler);** şeklindedir. Mod kısmına aşağıda verilen tanımlamalardan biri yazılır.

T2_DISABLED	: Timer2'yi kapat
T2_DIV_BY_1, T2_DIV_BY_4, T2_DIV_BY_16	: Prescaler bölme oranları

Periyot kısmına 0-255 arası bir değer girilir ve zamanlayıcı bu değere ulaştığında kendini sıfırlar.

Postscaler kısmına 1-16 arasında değer girilir. Bu değer kesme oluşması için Timer2'nin kaç defa sıfırlanması gerektiğini ifade eder.

2.6.3.2. SET_TIMER2( ) Komutu

Timer2 biriminin saymaya başlayacağı değeri ayarlar. Değer kısmına 0-255 arası bir değer yazılabilir. Kullanımı; **set_timer2(deger);** şeklindedir.

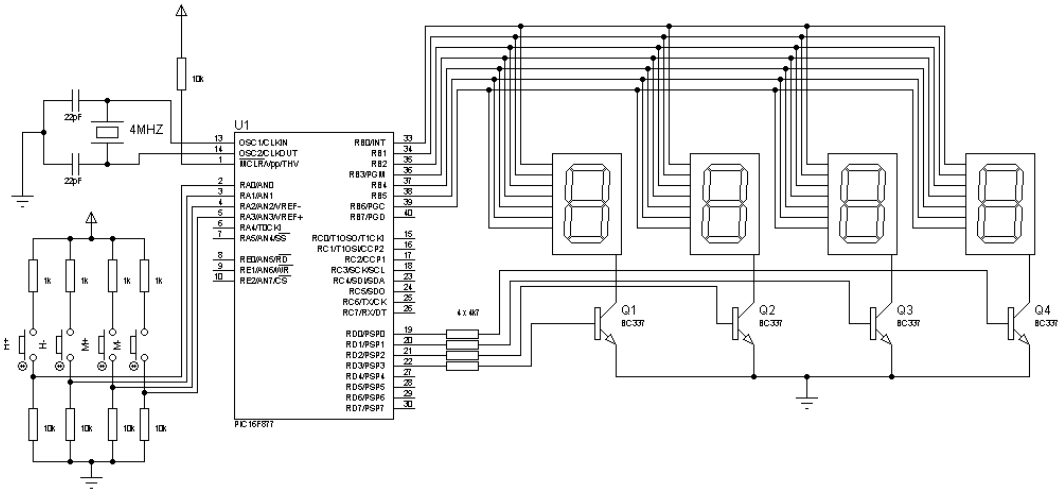
2.6.3.3. Timer2 Kesmesi

Kesmeler konusunda anlatıldığı gibi kullanılır. "#int_kesme adı" yerine "#int_timer2" yazılır. Kesmenin oluşma süresi;

Süre = $\frac{fosc}{4}$ x Bölme oranı x (peryot+1) x (postscaler değer) formulü ile hesaplanır.

2.7. Kesme ve Timer ile İlgili Örnek Yazılım Geliştirme

7 Segment displaylerle oluşturulmuş ve tarama yöntemi ile görüntü oluşturan saat devresi aşağıda verilmiştir. Bu devreye ait programı timer1 zamanlayıcısını ve timer1 kesmesini kullanarak yapalım.



Şekil 2.13: Ayarlanabilir saat devresi

DERLEYİCİ C Programı:

```
#include "D:\saat\SAAT-TİMER1.h"
#use fast_io(a)
#use fast_io(b)
#use fast_io(d)

int saat=0;
int dakika=0;
int saniye=0;
int m1,m10,h1,h10; // dakika ve saatin birler ve onlar basamağını gösterecek değişkenler
int tur=0;
const int dizi[10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f}; // display'de
                                                    // gösterilecek rakam bilgileri (0-9) tanımlanıyor.
#int_timer1      // Timer1 kesmesi
void timerkesmesi()      // Kesme oluştuğunda işlenecek fonksiyon başlangıcı
{
      set_timer1(3035);  // kesme içinde timer1 başlangıç değeri ayarlanıyor.
      tur++;              // her tur 0,5 saniyede gerçekleştiğinden,
        if(tur==2)                // 2 tur olduğunda 1 saniye olur.
        {
            tur=0;            // tur değeri sıfırlanıyor.
            saniye++;                // saniye değişkeni 1 arttırılıyor.
                if(saniye==60)      // saniye 60 olduysa,
            {
            dakika++;       // dakikayı 1 arttırır ve
            saniye=0;       // saniyeyi sıfırlar.
            }
                if(dakika==60)      // dakika 60 olduysa
```

```
            {
            saat++;                 // saati 1 arttırır ve
            dakika=0;       // dakikayı sıfırlar.
            }
                m1=dakika%10;   // dakika değerinin birler basamağını hesaplayıp m1'e atar.
                m10=dakika/10;  // dakika değerinin onlar basamağını hesaplayıp m10'a atar.
                h1=saat%10;     // saat değerinin birler basamağını hesaplayıp h1'e atar.
                h10=saat/10;    // saat değerinin onlar basamağını hesaplayıp h10'a atar.
        }

} // Kesme fonksiyonu sonu
void main()     // Ana fonksiyon başlangıcı
{
  setup_adc_ports(NO_ANALOGS);
  setup_adc(ADC_OFF);
  setup_psp(PSP_DISABLED);
  setup_spi(SPI_SS_DISABLED);
  setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
  setup_timer_2(T2_DISABLED,0,1);

set_tris_a(0x0f);         // A portunun ilk 4 pini giriş (butonlar bağlı)
set_tris_b(0x00);         // B portu komple çıkış
set_tris_d(0x00);         // D portu komple çıkış
  output_b(0x00);         // B portu lojik-0 yapılıyor.
  output_d(0x00);         // D portu lojik-0 yapılıyor.

  setup_timer_1(T1_INTERNAL|T1_DIV_BY_8); // Timer1 ayarlanıyor.
set_timer1(3035);         // Timer1 başlangıç değeri ayarlanıyor (0,5 sn için)
  enable_interrupts(int_timer1);          // timer1 kesmesi aktif ediliyor.
  enable_interrupts(GLOBAL); // aktif edilen kesmelere izin veriliyor.

  while(1)        // sonsuz döngü
{
    output_b(dizi[m1]); // dakika bilgisinin birler basamağına ait bilgi gönderiliyor.
    output_d(0x01);               // ilgili display aktif ediliyor.
    delay_ms(10);                 // 10 ms gecikme veriliyor.
    output_d(0x00);               // tüm displayler pasif (görüntü bozulmasını önlemek için)

    output_b(dizi[m10]);          // dakika bilgisinin onlar basamağına ait bilgi gönderiliyor.
    output_d(0x02);               // ilgili display aktif ediliyor.
    delay_ms(10);                 // 10 ms gecikme veriliyor.
    output_d(0x00);               // tüm displayler pasif (görüntü bozulmasını önlemek için)

    output_b(dizi[h1]);  // saat bilgisinin birler basamağına ait bilgi gönderiliyor.
    output_d(0x04);               // ilgili display aktif ediliyor.
    delay_ms(10);                 // 10 ms gecikme veriliyor.
```

```
    output_d(0x00);             // tüm displayler pasif (görüntü bozulmasını önlemek için)

    output_b(dizi[h10]);// saat bilgisinin onlar basamağına ait bilgi gönderiliyor.
    output_d(0x08);             // ilgili display aktif ediliyor.
    delay_ms(10);               // 10 ms gecikme veriliyor.
    output_d(0x00);             // tüm displayler pasif (görüntü bozulmasını önlemek için)

    if(input(pin_a0)==1)        // A0 pinine bağlı buton basıldıysa
{
      delay_ms(20);             // arkları önlemek için gecikme
saat++;                   // saat değişken değerini 1 arttır.
      while(input(pin_a0)==1); // buton bırakılıncaya kadar bekle.
}
    if(input(pin_a1)==1)        // A1 pinine bağlı buton basıldıysa
{
      delay_ms(20);             // arkları önlemek için gecikme.
saat--;                   // saat değişken değerini 1 azalt.
      while(input(pin_a1)==1); // buton bırakılıncaya kadar bekle.
}
    if(input(pin_a2)==1)        // A2 pinine bağlı buton basıldıysa
{
      delay_ms(20);             // arkları önlemek için gecikme.
dakika++;                 // dakika değişken değerini 1 arttır.
      while(input(pin_a2)==1); // buton bırakılıncaya kadar bekle.
}
    if(input(pin_a3)==1)        // A3 pinine bağlı buton basıldıysa
{
      delay_ms(20);             // arkları önlemek için bekle.
dakika--;                     // dakika değişken değerini 1 azalt.
      while(input(pin_a3)==1); // buton bırakılıncaya kadar bekle.
}
} // Sonsuz döngü sonu
} // ana fonksiyon sonu
```

2.8. Capture/Compare/Pwm Biriminin Çalıştırılması

2.8.1. Capture (Yakalama) Modu

Denetleyicinin capture modunda belirlenen pininden gelen clock sinyali belirtilen şarta göre değerlendirilir. Belirtilen şart sağlandığında yakalama olayı gerçekleşmiş olur. TMR1 kaydedici içeriği 8'er bitlik CCPRxH ve CCPRxL kaydedicilere aktarılır. Bu kaydedicilerden CCPRxH, 16 bitlik Timer1 kaydedici içeriğinin yüksek değerlikli 8 bitini, CCPRxL ise düşük değerlikli 8 bitini tutar. DERLEYİCİ C de CCPRxH kaydedicisi yerine

CCP_x_HIGH, CCPRxL yerine ise CCP_x_LOW kullanılır ve x yerine kullanılan CCP modül numarası (1 veya 2) yazılır.

2.8.2. Compare (Karşılaştırma) Modu

CCPRx kaydedicisi içindeki değer ile TMR1 kaydedicisi içindeki değer eşit olduğunda CCPx kesmesi oluşur. Burada x yerine CCP modül numarası (1 veya 2) yazılır.

2.8.3. PWM Modu

CCP modülü PWM olarak kullanıldığında compare ve capture den farklı olarak Timer2 zamanlayıcısını kullanır. PWM sinyalinin "**set_pwmX_duty(deger)**;" koduyla belirtilen görev çevrim süresi (deger), Timer2 ayarlamasında belirtilen periyot değerinden hiçbir zaman büyük olmamalıdır.

```
set_pwm1_duty(60);
```

2.8.4. SETUP_CCPx( ) Komutu

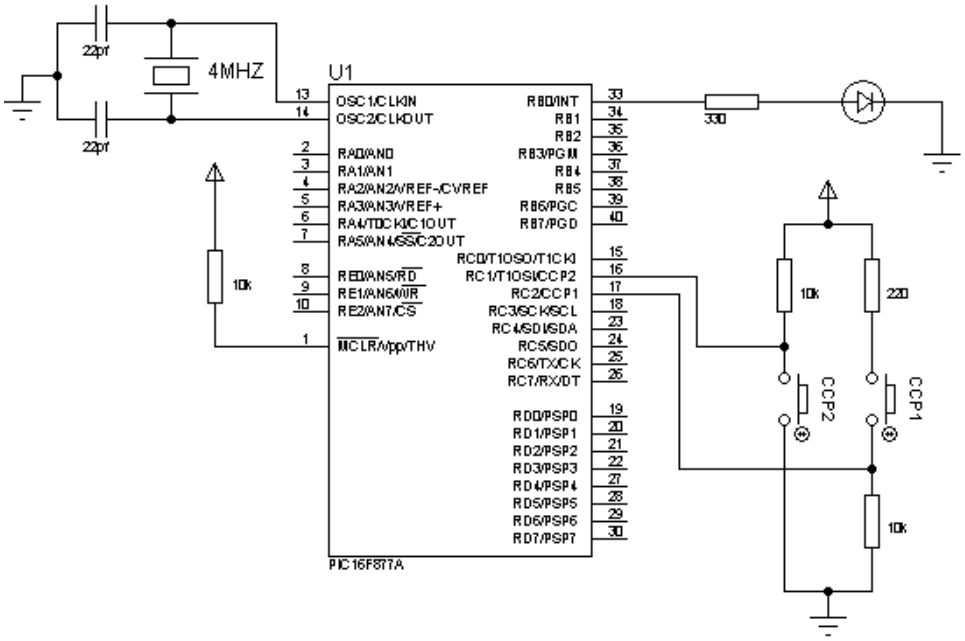
Denetleyici içinde bulunan CCP modülü ile ilgili ayarları yapan fonksiyondur. Kullanımı;
setup_ccpX(mod);şeklindedir, Buradaki X yerine kullanılacak CCP modül numarası (1 veya 2) yazılır. Mod yerine ise aşağıda belirtilen ifadelerden biri yazılır.

CCP_OFF : CCP modülü kapalı.
CCP_CAPTURE_FE : Her düşen kenarda yakala (capture modu).
CCP_CAPTURE_RE : Her yükselen kenarda yakala (capture modu).
CCP_CAPTURE_DIV_4 : Her 4.yükselen kenarda yakala (capture modu).
CCP_CAPTURE_DIV_16 : Her 16.yükselen kenarda yakala (capture modu)
CCP_COMPARE_SET_ON_MATCH : Eşleşme olduğunda CCPx pini 1 olsun ve kesme meydana gelsin (compare modu)
CCP_COMPARE_CLR_ON_MATCH : Eşleşme olduğunda CCPx pini 0 olsun ve kemse meydana gelsin (compare modu)
CCP_COMPARE_INT : Eşleşme olduğunda sadece kesme oluşsun.
CCP_COMPARE_RESET_TIMER : Eşleşme olduğunda kesme oluşsun ve Timer1 Sıfırlansın
CCP_PWM : PWM modunu aç.

setup_ccp1(CCP_COMPARE_INT);

Örnek: CCP Modül Uygulaması

Aşağıda verilen devrede 16f877A denetleyicisinin CCP modülü harici sinyal giriş uçları olan RC1 ve RC2 uçlarına iki adet buton bağlanmıştır. RC2 ye bağlı butona basıldığında LED'i yakan ve RC1 e bağlı butona basıldığında LED'i söndüren devreyi capture özelliğini kullanarak yapalım.



Şekil 2.14: Capture modu uygulama devresi

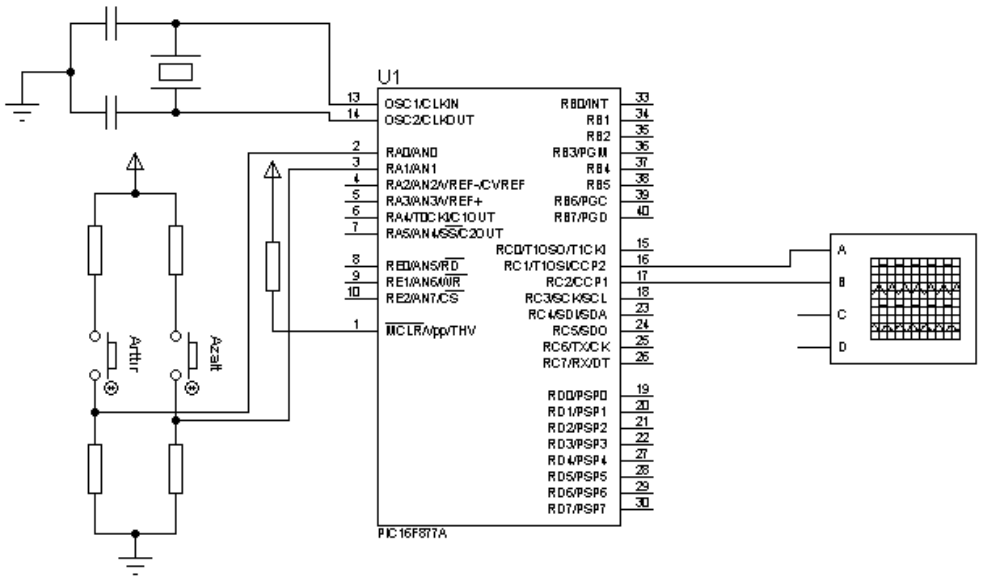
DERLEYİCİ CProgramı:

```
#include "D:\\ISIS\capture.h"
#use fast_io(b)          // B portu giriş çıkış olarak kullanılacak.
#use fast_io(c)          // C portu giriş çıkış olarak kullanılacak.

#INT_CCP1                // CCP1 kesmesi
   void ccp1_kesmesi()
   {
     output_high(pin_b0);        // B0 pinine bağlı LED'i yakar.
   }
#INT_CCP2                // CCP2 kesmesi
   void ccp2_kesmesi()
   {
     output_low(pin_b0);         // B0 pinine bağlı LED'i söndürür.
   }
void main()      // Ana fonksiyon başlangıcı
{
```

```
  setup_adc_ports(NO_ANALOGS);
  setup_adc(ADC_OFF);
  setup_psp(PSP_DISABLED);
  setup_spi(SPI_SS_DISABLED);
  setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
  setup_timer_1(T1_DISABLED);
  setup_timer_2(T2_DISABLED,0,1);
  setup_comparator(NC_NC_NC_NC);
  setup_vref(FALSE);

set_tris_b(0x00);        // B portu çıkış olarak yönlendiriliyor.
set_tris_c(0x06);        // C portunun C1 ve C2 pinleri giriş, diğerleri çıkış olarak
yönlendiriliyor.
  output_b(0x00);        // B portu çıkışı sıfırlanıyor.

  enable_interrupts(INT_CCP1);        // CCP1 kesmesi aktif ediliyor.
  enable_interrupts(INT_CCP2);        // CCP2 kesmesi aktif ediliyor.
  enable_interrupts(GLOBAL);          // Aktif edilen kesmelere izin veriliyor.

  setup_ccp1(CCP_CAPTURE_RE);   // CCP1, her yükselen kenarda yakalasın.
  setup_ccp2(CCP_CAPTURE_FE);   // CCP2, her düşen kenarda yakalasın.

  CCP_1_HIGH=0x00; // Kaydedici içerikleri sıfırlanıyor.
  CCP_1_LOW=0x00;  //        "        "        "
  CCP_2_HIGH=0x00; //        "        "        "
  CCP_2_LOW=0x00;  //        "        "        "

  while(1);       // Sonsuz döngü
}
```

2.9. PWM Uygulamaları

2.9.1. PWM Sinyalin Görev Çevrim Süresini Değiştirme



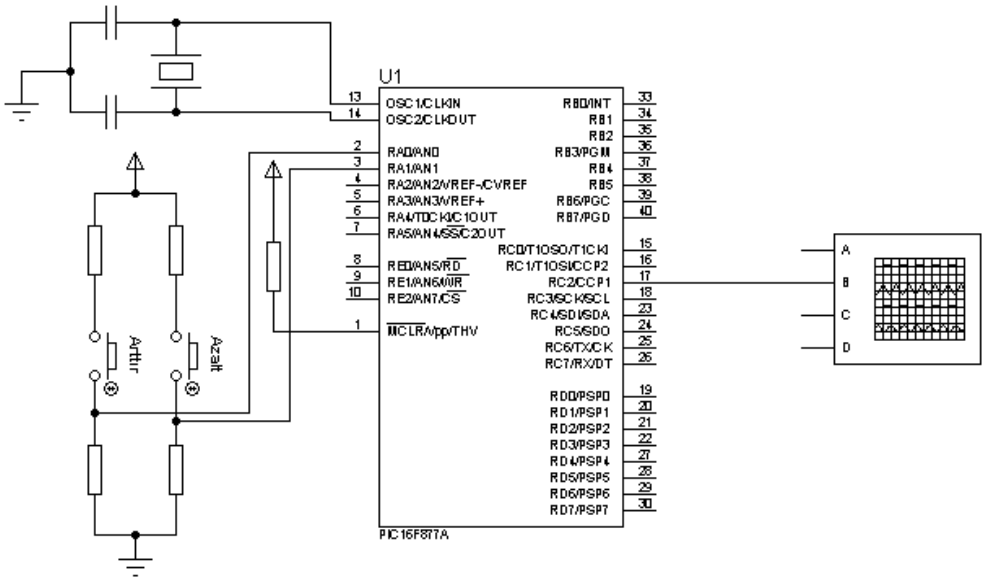
Şekil 2.15: PWM uygulama devresi

DERLEYİCİ C Programı:

```
#include "D:\ISIS\PWM_Duty_Cycle.h"

#use fast_io(a)
#use fast_io(c)
int i=5;

void main()
{
  setup_adc_ports(NO_ANALOGS);
  setup_adc(ADC_OFF);
  setup_psp(PSP_DISABLED);
  setup_spi(SPI_SS_DISABLED);
  setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
  setup_timer_1(T1_DISABLED);

set_tris_a(0x03); // RA0 ve RA1 giriş yapılıyor.
set_tris_c(0x00); // C portu çıkış yapılıyor.

  setup_ccp1(CCP_PWM); // CCP1 birimi PWM için ayarlandı.
  setup_ccp2(CCP_PWM); // CCP2 birimi PWM için ayarlandı.

  setup_timer_2(T2_DIV_BY_16,170,1); // Timer2 ayarlandı.
```

```
set_pwm1_duty(i); // PWM1 çıkışı görev çevrimi belirleniyor.
set_pwm2_duty(i); // PWM2 çıkışı görev çevrimi belirleniyor.
  while(1)
{
    if(input(pin_a0))
{
      delay_ms(20);
      while(input(pin_a0)); // Butondan el çekilinceye kadar bekle.
i+=3;
      if(i>=170)
i=170;
set_pwm1_duty(i); // PWM1 çıkışı görev çevrimi belirleniyor.
set_pwm2_duty(i); // PWM2 çıkışı görev çevrimi belirleniyor.
}
    if(input(pin_a1))
{
      delay_ms(20);
      while(input(pin_a1)); // Butondan el çekilinceye kadar bekle.
i-=3;
      if(i<=5)
i=5;
set_pwm1_duty(i); // PWM1 çıkışı görev çevrimi belirleniyor.
set_pwm2_duty(i); // PWM2 çıkışı görev çevrimi belirleniyor.
}
}
}
```

2.9.2. PWM Sinyalin Frekansını Değiştirme



Şekil 2.16: PWM Uygulama Devresi

DERLEYİCİ C Programı:

```
#include "C:\ISIS\PWM_Frekans.h"

#use fast_io(a)
#use fast_io(c)

int i=65;
void main()
{
  setup_adc_ports(NO_ANALOGS);
  setup_adc(ADC_OFF);
  setup_psp(PSP_DISABLED);
   setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);

set_tris_a(0x03); // RA0 ve RA1 giriş olarak ayarlanıyor.
set_tris_c(0x00); // C portu çıkış olarak ayarlanıyor.

  setup_ccp1(CCP_PWM); // CCP1 birimi PWM çıkışı için ayarlanıyor.

  setup_timer_2(T2_DIV_BY_16,i,1); // Timer2 ayarlanıyor.

set_pwm1_duty(50); // PWM1 görev çevrimi belirleniyor.

  while(1)
{
    if(input(pin_a0))
```

```
{
        delay_ms(20);
        while(input(pin_a0)); // Buton bırakılıncaya kadar bekle.
i+=5;
        if(i>=250)
i=250;
        setup_timer_2(T2_DIV_BY_16,i,1); // Timer2 ayarlanıyor.
}
     if(input(pin_a1))
{
        delay_ms(20);
        while(input(pin_a1)); // Buton bırakılıncaya kadar bekle.
i-=5;
        if(i<=65)
i=65;
        setup_timer_2(T2_DIV_BY_16,i,1); // Timer2 ayarlanıyor.
}
}
}
```

UYGULAMA FAALİYETİ

PWM sinyalinin görev çevrim süresini değiştirme uygulaması yapınız.

Kullanılan araç ve gereçler:
- Bilgisayar
- Mikrodenetleyici programlama kartı
- Mikrodenetleyici deney seti ve osiloskop

İşlem Basamakları	Öneriler
➢ Yeni bir proje dosyası oluşturunuz. ➢ Deney setinizi inceleyerek Şekil 2.15'de verilen devreye uygun hale getiriniz. ➢ Yapacağınız programa ait akış şemasını oluşturunuz. ➢ Akış şemasından faydalanarak kodlarınızı yazınız. ➢ Programınızı öğretmeniniz ile birlikte inceleyiniz. ➢ Programınızı derleyiniz. ➢ Derleme sonucu oluşan. hex dosyasını öğretmeniniz gözetiminde denetleyiciye yükleyiniz. ➢ Deney setinde yazdığınız programın çalışmasını osiloskop yardımıyla inceleyiniz.	➢ Proje dosyasını PWM uygulaması isminde bir klasör içine Duty-Cycle ismiyle oluşturunuz. ➢ Butonlar farklı port ya da pinlere de bağlı olabilir. Böyle bir durum varsa program kod alanında ilgili pinler için gerekli değişiklikleri yapınız. ➢ Akış şeması oluşturmada sıkıntı yaşarsanız öğretmeninizden yardım isteyiniz. ➢ Takıldığınız yerde bölüm 2.9.1'de verilen DERLEYİCİ C kodlarından yararlanabilirsiniz. ➢ Derleme işlemi Compile menüsü altında yer alan Compile komutu ile yapılır. ➢ PIC programlayıcınızın türüne bağlı olarak denetleyiciyi uygun şekilde takın ve varsa gerekli jumper ayarlarını yapınız. ➢ Osiloskop kullanmayı bilmiyorsanız öğretmeninizden yardım isteyebilirsiniz.

KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız becerileri **Evet**, kazanamadığınız becerileri **Hayır** kutucuğuna (**X**) işareti koyarak kendinizi değerlendiriniz.

Değerlendirme Ölçütleri	Evet	Hayır
1. Yeni bir proje dosyası oluşturabildiniz mi?		
2. Deney setinde gerekli değişiklikleri yapabildiniz mi?		
3. Programın akış şemasını çizebildiniz mi?		
4. DERLEYİCİ C dilini kullanarak programı yazabildiniz mi?		
5. Programı derleyebildiniz mi?		
6. Programı denetleyiciye yükleyebildiniz mi?		

DEĞERLENDİRME

Değerlendirme sonunda "Hayır" şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınız "Evet" ise "Ölçme ve Değerlendirme"ye geçiniz.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki cümleleri dikkatlice okuyarak boş bırakılan yerlere doğru sözcüğü yazınız.

1. ……komutu denetleyici portlarının ve bu portlara ait pinlerin giriş/çıkış olarak ayarlanmasını sağlar.

2. …….komutu ile bir portun tüm pinlerine bilgi gönderilir.

3. …….komutu ile bir portun istenen pini lojik-0 yapılır.

4. …….komutu ile istenen pinin değeri terslenir.

5. Giriş olarak ayarlanan bir portun değeri ……komutu ile okunur.

6. Giriş olarak ayarlanan bir pinin değeri …….komutu ile okunur.

7. Bir değişkenin birden fazla değer alma ihtimalinin olduğu yerlerde karar komutu olarak …….komutu tercih edilir.

8. Bir döngünün en az birkere işlenmesi gereken durumlarda …….döngüsü kullanılır.

9. Displayler ……sütun ve ……sütun olmak üzere iki çeşittir.

10. Giriş-çıkış işlemlerinin denetleyicinin donanımsal özellikleri ile yapılmasına ……..denir.

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise “Modül Değerlendirme”ye geçiniz.

MODÜL DEĞERLENDİRME

Aşağıdaki cümleleri dikkatlice okuyarak boş bırakılan yerlere doğru sözcüğü yazınız.

1. ……..ANSI C'nin temel tanımlamalarının büyük çoğunluğunu kabul eder ve diğer C temelli mikrodenetleyici derleyicilerine göre C dili ile daha fazla uyum içindedir.

2. Otomatik proje dosyası oluşturmak için ……menüsü altında yer alan ……komutu kullanılır.

3. Manuel olarak proje dosyası oluşturmak için …….menüsü altında yer alan …….komutu kullanılır.

4. Yazılan program; ……..menüsü altında yer alan ……..komutu ile derlenir.

5. Yazılan programa akış diyagramı ve elektriksel devre oluşturmak için …….menüsü altında yer alan …….komutu kullanılır.

6. ……. komutu ile bir portun tüm pinlerine bilgi gönderilir.

7. ……. komutu ile istenen pinin değeri terslenir.

8. Giriş olarak ayarlanan bir pinin değeri …….. komutu ile okunur**.**

9. Bir döngünün en az bir kere işlenmesi gereken durumlarda ……. döngüsü kullanılır.

10. Giriş-çıkış işlemlerinin denetleyicinin donanımsal özellikleri ile yapılmasına …….. denir.

DEĞERLENDİRME

Değerlendirme sonunda **"Hayır"** şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınız **"Evet"** ise bir sonraki modüle geçmek için öğretmeninize başvurunuz.

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ-1'İN CEVAP ANAHTARI

1	Assembly – Günlük İngilizce diline
2	DERLEYİCİ C derleyicisi
3	File - Open
4	Project – PIC Wizard
5	Project - Creat
6	Compile – C/ASM
7	Compile – Compile
8	Document – Flow Chart
9	-128 ile +127
10	3

ÖĞRENME FAALİYETİ-2'NİN CEVAP ANAHTARI

1	set_tris_X( )
2	output_X( )
3	output_low(pin adı)
4	output_toggle(pin adı)
5	input_X( )
6	input(pin adı)
7	switch-case( )
8	do-while( )
9	Anot - Katot
10	kesme

MODÜL DEĞERLENDİRMENİN CEVAP ANAHTARI

1	**DERLEYİCİ C derleyicisi**
2	**Project - PIC Wizard**
3	**Project - Creat**
4	**Compile - Compile**
5	**Document - Flow Chart**
6	**output_X( )**
7	**output_toggle(pin adı)**
8	**input(pin adı)**
9	**do-while( )**
10	kesme

KAYNAKÇA

- ÇİÇEK Serdar, **DERLEYİCİ C ile PIC Programlama**, Altaş Yayıncılık, Mart, 2009.
- DEVECİ Murat, **Pic Programlama**, Ağustos, 2009.
- UYSAL Mithat, **C ile programlama- Beta**, Kasım, 2001.
- ZENGIN F., M.Şanlı, O.Urhan, M.K Güllü, **C Dili ile PIC Uygulamaları**, Birsen Yayınevi, 2006.